

В ПОМОЩЬ РАДИОЛЮБИТЕЛЮ

Патрик Гёлль

Как превратить персональный компьютер в универсальный программатор

Микросхемы ПЗУ, ПЛМ, ПЛИС и приставки для их программирования

Программируемые запоминающие устройства

Программируемые логические схемы

Микроконтроллеры

Программаторы

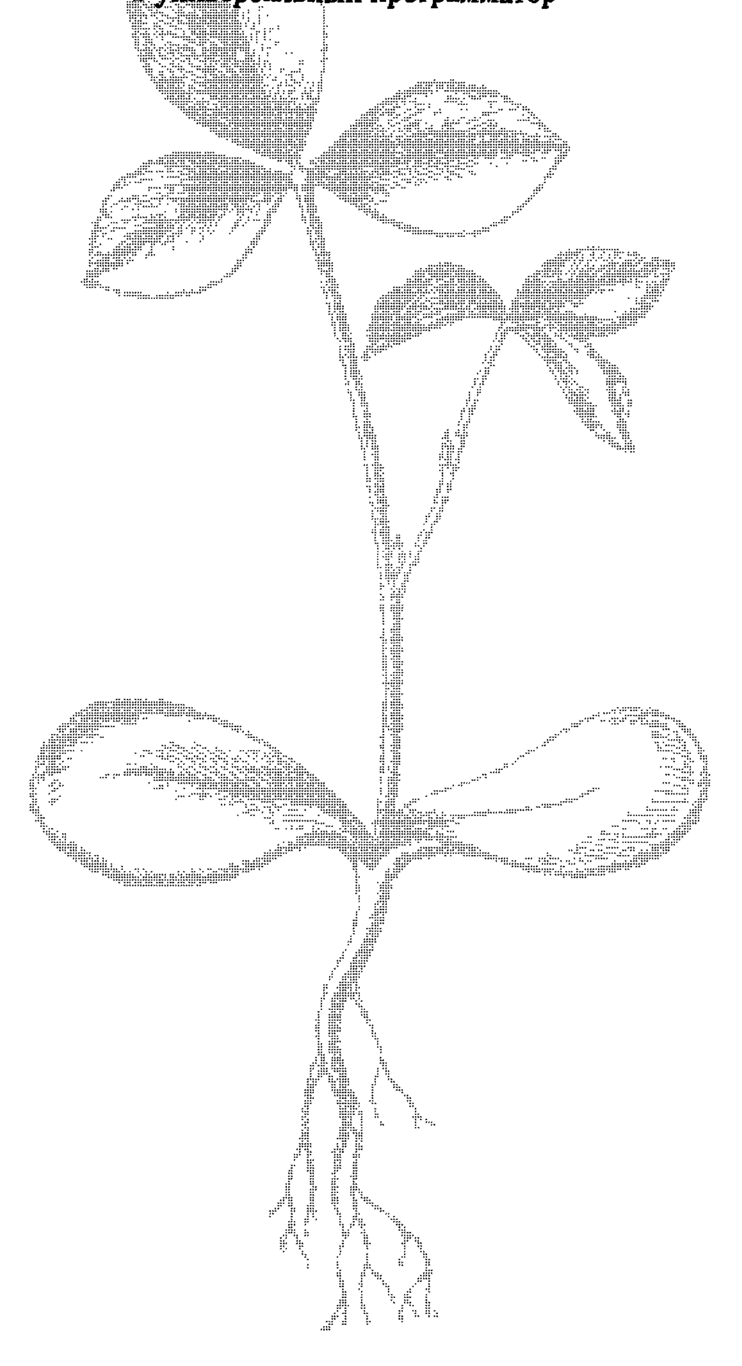
ETSF

DUNOD

ДМК

Патрик Гёлль

Как превратить персональный компьютер в универсальный программатор



Patrick Gueulle

COMPOSANTS ÉLECTRONIQUES PROGRAMMABLES SUR PC





EDITIONS TECHNIQUES ET SCIENTIFIQUES FRANÇAISES

В помощь радиолюбителю

Патрик Гёлль

КАК ПРЕВРАТИТЬ ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР В УНИВЕРСАЛЬНЫЙ ПРОГРАММАТОР



Москва, 2000

ББК 32.844-я92
Г31

Гёлль П.

Г31. Как превратить персональный компьютер в универсальный программатор: Пер. с франц. – М.: ДМК, 2000. – 168 с.: ил. (В помощь радиолюбителю).

ISBN 5-93700-017-X

В новой книге известного французского инженера и радиолюбителя рассматриваются наиболее распространенные типы самых современных интегральных микросхем – многократно перепрограммируемых. Представлены все основные классы: ИМС памяти, программируемые логические ИМС и микроконтроллеры. Описаны простые и надежные программаторы, приведены программы для управления ими, рассмотрены программные комплексы и системы разработки для ПЛИС и микроконтроллеров.

Издание предназначено для инженеров, радиолюбителей и студентов вузов, желающих познакомиться с новыми классами активных электронных компонентов.

ББК 32.844-я92

ISBN 2-10-023987-2 (франц.)
ISBN 5-93700-017-X (рус.)

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Изготовление собственных интегральных микросхем – это мечта, которую, вероятно, хоть раз в жизни лелеял каждый специалист по электронике. Техническая мысль все ближе к этой мечте: программируемые электронные радиоэлементы позволяют выполнять многочисленные задачи, которые еще несколько лет назад были практически неразрешимы.

Но работа с этими радиоэлементами требует использования определенных методов и специального оборудования, ведь и речи быть не может о том, чтобы орудовать паяльником внутри интегральной микросхемы! Для правильной настройки программируемого радиоэлемента, который в дальнейшем можно применять как обычную интегральную микросхему, в первое время придется прибегнуть к помощи клавиатуры компьютера, а затем, чтобы «зашить» в кристалл уже подготовленную схему, вам понадобится специальный прибор – программатор.

Эта книга была задумана не только ради знакомства читателя с технологией программируемых радиоэлементов, но и для того, чтобы помочь начинающему радиолюбителю избежать значительных затрат, которые предполагаются в рекомендациях некоторых фирм-изготовителей или крупных поставщиков.

Мы не ставили перед собой цель описать все существующие варианты и типы программируемых радиоэлементов, номенклатура которых к тому же постоянно расширяется. Здесь не будет описания тех изделий, процедуры программирования которых засекречены. Мы предпочли отобрать наиболее распространенные – те, которые можно легко приобрести по приемлемым ценам.

Необходимый для работы программатор может быть собран без существенных затрат или приобретен в составе «набора для начинающих» (Starter Kit). Тем, кто решит заниматься изучением более сложных программируемых радиоэлементов, потребуются «универсальные» программаторы, стоимость которых значительно выше.

ГЛАВА СТРАНИЦА

1 ВВЕДЕНИЕ В МИР ПРОГРАММИРУЕМЫХ КОМПОНЕНТОВ

ЭВОЛЮЦИЯ ЭЛЕКТРОННЫХ СХЕМ

В современной электронике все большее значение приобретают интегральные микросхемы, пришедшие на смену дискретным радиодеталям, которые, однако, все еще встречаются в некоторых областях электроники, например там, где требуется особая точность. Кроме того, обойтись только стандартными интегральными микросхемами невозможно, и использование некоторого количества дискретных элементов просто необходимо, чтобы собираемая схема полностью отвечала предъявляемым к ней требованиям.

Наиболее заметной тенденцией последних лет является быстрый рост интеграции микросхем. Благодаря непрерывно совершенствующимся технологиям одна интегральная микросхема в настоящее время может содержать эквивалент микро-ЭВМ средней мощности.

Сфера применения цифровых методов обработки информации постоянно расширяется, они все чаще вытесняют использовавшиеся ранее аналоговые схемы, позиции которых тем не менее все еще непоколебимы в некоторых областях науки и техники.

Наконец, программное обеспечение становится все более необходимой частью любого проекта, при этом затраты на аппаратное обеспечение постоянно снижаются. Часто именно в программе микропроцессора скрыты все секреты устройства, которое с точки зрения схемотехники может выглядеть весьма банально, тем не менее скопировать и размножить программу гораздо легче, чем правильно повторить топологию и изготовить печатные платы, пусть даже несложные. Таким образом, вопрос о защите производственных секретов остается постоянно открытым.

ЭВОЛЮЦИЯ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

Новые интегральные микросхемы становятся все более специализированными. Если одна и та же микросхема с четырьмя двухвходовыми элементами И-НЕ может использоваться в самых разных схемах, то сказать это же о видеоконтроллере для компьютерного дисплея или о синтезаторе звука для «говорящих» игрушек нельзя.

Несмотря на огромный выбор специализированных изделий, представленных на рынке, разработчик не всегда может найти тот единственный чудодейственный компонент, который поможет решить все его проблемы. И даже если компонент этот найден, то всегда надо учитывать и возможность задержки в поставках, и отсутствие так называемых «вторых поставщиков» (изготовителей аналогичных компонентов по лицензии основного производителя) и, самое главное – вероятность прекращения его производства.

Чрезвычайно опасно ставить весь разрабатываемый проект в зависимость от одного радиоэлемента, поставляемого единственным производителем. Отсутствие конкурентов или «вторых поставщиков» может в любое время вызвать скачок цен. Кроме того, всегда существует риск, что через несколько месяцев производство необходимого изделия прекратится без всякого предварительного предупреждения и, естественно, без возмещения ущерба.

В этом случае продолжать свое производство можно, пока не кончатся складские запасы, и одновременно готовиться к серьезным проблемам, связанным с техническим и гарантийным обслуживанием уже изготовленного и поставленного заказчикам оборудования.

В связи с вышесказанным возникает вопрос: разумно ли использовать только «универсальные» стандартизированные радиоэлементы? Очевидно, нет, так как их универсальность требует применения большого количества дискретных деталей, значительно усложняющих схему и конструкцию проектируемых устройств.

ИНТЕГРАЛЬНЫЕ МИКРОСХЕМЫ ЧАСТНОГО ПРИМЕНЕНИЯ

В тех случаях, когда крупносерийное производство каких-либо изделий и систем гарантировано на длительный период, наиболее эффективным будет использование производимых в заводских условиях «заказных» или «полузаказных» интегральных микросхем (Custom IC или ASIC – Application Specific Integrated Circuit). Это весьма хлопотное капиталоемкое и рискованное решение, которое, однако, исключительно рентабельно в случае, если речь идет о сотнях тысяч единиц.

Первоначальные инвестиции при разработке заказных БИС очень велики, но оправданы: вы получите компонент, наиболее полно отвечающий конкретным техническим требованиям. Он не будет выпускаться в продажу, и его будет практически невозможно скопировать. И наоборот, любая модификация потребует повторения длинной и дорогостоящей процедуры разработки БИС. «Права на ошибку» не существует, а в процессе серийного производства практически отсутствует возможность модернизации конечного продукта, в котором использованы «заказные» микросхемы. В любом случае, подобное решение практически неприменимо для мелких и средних объемов выпуска и особенно для опытных или единичных изделий или образцов.

ПРОГРАММИРУЕМЫЕ ИНТЕГРАЛЬНЫЕ МИКРОСХЕМЫ

Понятие «программируемая логика» хорошо известно и представляет собой полную противоположность «логике жесткой». В устройствах на жесткой логике поведение схемы полностью определяется

связями между определенным числом элементов. Модернизация или изменение схемы устройства достаточно трудоемки, а для ее копирования вполне достаточно определенного терпения.

В системах с программируемой логикой используется центральный процессор (микропроцессор, микроконтроллер), который связан с какими-либо периферийными устройствами и выполняет программу, содержащуюся в энергонезависимой памяти той или иной конструкции. Именно программа в значительной степени определяет поведение всей системы. Все изменения и модификации программного обеспечения вносятся в систему путем перезагрузки центрального процессора, при этом никоим образом не затрагивается аппаратная часть устройства, то есть его электрическая схема. Одна и та же система может применяться для управления котлом теплоснабжения или использоваться в устройствах автоматизированной телефонной связи. Все зависит от программного обеспечения, загруженного в ее память.

Устройство, в котором центральный процессор и его периферийные устройства (особенно блоки памяти) объединены в одном корпусе, называется микроконтроллером, или «однокристалкой» (ОМЭВМ – однокристальная микро-ЭВМ); оно может быть запрограммировано для применения в самых разных системах и областях, а также для решения различных задач. Однокристальные микро-ЭВМ – самые главные и самые дорогие части таких систем. Разработка технологических и других способов защиты от незаконного копирования «зашитого» в них программного обеспечения надежно застраховала их от подделок. Это значительный шаг вперед.

Существуют устройства, занимающие промежуточное положение между жесткой логикой и однокристальными микро-ЭВМ. Речь идет о программируемых логических интегральных схемах (ПЛИС), которые очень популярны у производителей. Это не программируемые запоминающие устройства (ПЗУ), а интегральные микросхемы, внутренняя структура которых может многократно изменяться самим пользователем, причем сделать это очень легко.

Теперь можно самостоятельно «произвести» оригинальные радиоэлементы либо в единственном экземпляре, либо в виде небольшой партии, причем без огромных капиталовложений, необходимых для производства «заказных» микросхем. Программируемые логические ИС давно известны профессиональным разработчикам РЭА и сейчас начинают более широко применяться квалифицированными и подготовленными радиолюбителями. Главная задача этой книги состоит в том, чтобы существенно расширить круг людей, использующих самые современные технологии.

Запоминающие устройства необходимы в самых различных областях электроники, их применение совсем не ограничено сферой информатики.

Улучшение технологий изготовления запоминающих устройств и взрывное повышение спроса на них привели к снижению цен, увеличению емкости, а также к появлению новых разновидностей, с помощью которых можно реализовать приложения, казавшиеся всего несколько лет назад невероятными.

Особое место в этом семействе компонентов занимают запоминающие устройства, которые можно обозначить как «программируемые».

СЕМЕЙСТВА ПРОГРАММИРУЕМЫХ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Отдельные разновидности запоминающих устройств часто идентифицируют по первым буквам в их английских названиях, поэтому значения таких аббревиатур важно хорошо знать.

Отметим сначала оперативные запоминающие устройства (ОЗУ) и постоянные запоминающие устройства (ПЗУ). В ОЗУ можно записывать данные и считывать их, но хранящаяся в них информация теряется после отключения напряжения питания. ПЗУ используются главным образом в режиме считывания, и данные находятся в них в более или менее устойчивом состоянии.

Оперативные запоминающие устройства (ОЗУ) в иностранной литературе обычно называют RAM – Random Access Memories (запоминающие устройства с произвольной выборкой, ЗУПВ). Постоянные запоминающие устройства (ПЗУ) называют ROM – Read Only Memories.

ОЗУ делятся на две категории: динамические (DRAM) и статические (SRAM). Динамические ОЗУ намного проще в изготовлении и могут иметь емкости памяти, исчисляющиеся мегабитами. Они намного дешевле, чем статические ОЗУ, но нуждаются в постоянном «освежении» (refreshing – регенерация) хранящихся в них данных, что заметно усложняет их использование.

В принципе ОЗУ не являются «программируемыми радиоэлементами» в том смысле, который обычно вкладывают в термин «радиоэлементы»: если пользователь тем или иным образом определил их содержимое, эта информация не должна изменяться в течение длительного времени даже при отключении питания. Но из данного правила есть исключения, о которых будет сказано ниже.

Что касается ПЗУ, то в них следует выделить постоянные запоминающие устройства, программируемые один раз, и перепрограммируемые с возможностью стирания (СППЗУ).

К первой категории принадлежат так называемые масочные ПЗУ, в которые данные заносятся на этапе изготовления микросхем, а также ППЗУ – программируемые ПЗУ (Programmable ROM) с пережигаемыми перемычками. Такие ППЗУ пользователь может программировать сам, но необратимо.

Наиболее известными перепрограммируемыми запоминающими устройствами являются СППЗУ (Erasable PROM, EPROM) полностью стираемые, например, при облучении их ультрафиолетовыми лучами (УФ СППЗУ, называемые иногда ЭППЗУ – электрически программируемые), перепрограммировать их можно сотни раз, но сейчас их выпускают также и в версии ОП – однократно программируемые (One Time Programmable, OTP). В таком исполнении у экономичного пластмассового корпуса нет прозрачного окошка и данные, записанные в ОП СППЗУ, стереть нельзя.

Микросхемы памяти, информацию в которых можно полностью или частично стереть чисто электрическим путем, часто «прозрачным» для пользователя, называются ЭСППЗУ – электрически стираемые и перепрограммируемые ПЗУ (Electrically Erasable Programmable ROM – EEPROM или E2PROM), которые являются интересной альтернативой УФ СППЗУ в тех случаях, когда необходимы частые модификации информации в ПЗУ.

Очень похожи на СППЗУ запоминающие устройства типа флэш (Flash), где стирание информации происходит также по всему объему одновременно, но не ультрафиолетовыми лучами, а электрически. Запоминающие устройства такого типа довольно широко используются в некоторых областях техники, но не способны полностью подменить обычные СППЗУ.

КМОП ОЗУ со встроенными литиевыми элементами питания сочетают в себе свойства и ОЗУ, и ПЗУ. Данные в них можно записывать, как в обычные ОЗУ, а затем использовать как ПЗУ, поэтому в целом они более удобны и производительны, чем ЭСППЗУ. Удобны они и в процессе разработки, отладки и подготовки производства.

ЧТО СОДЕРЖАТ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Являясь истинно цифровыми интегральными микросхемами, запоминающие устройства содержат информацию вида «да» или «нет»: 1 и 0, обычно называемые битами или двоичными разрядами. Естественно,

ничто не мешает принять соглашение и называть группы из некоторого числа битов тем или иным термином, как это принято для аналоговых запоминающих устройств, в частности в области синтеза речи.

Одной из главных характеристик запоминающего устройства вне зависимости от типа является его организация. Запоминающее устройство емкостью 64 Кбит, например, может быть организовано как 65536 слов по одному биту, как 8192 слов по восемь бит, как 16384 слов по четыре бита и т.д. (напомним, что на самом деле приставка «кило» в цифровой технике обозначает 1024 единицы).

Наиболее распространенной структурой запоминающих устройств издавна является организация данных в слова по восемь бит, иначе говоря, в слова длиной в байт (byte). Каждое слово может рассматриваться как ячейка в запоминающем устройстве, которая имеет индивидуальный номер, называемый адресом. Таким образом, можно, указав соответствующий адрес, считывать или записывать в каждую ячейку такого запоминающего устройства байты информации.

Если появляется необходимость использовать более длинные, чем восемь бит, слова, допустимо применить несколько запоминающих устройств параллельно. С двумя запоминающими устройствами побайтной организации, например, можно работать со словами длиной 16 бит.

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ

Двоичное представление

Наиболее простой способ определить содержание запоминающего устройства состоит в том, чтобы отображать непосредственно биты, которые оно содержит.

Слово из восьми бит, в частности, определяет 256 (два в восьмой степени или количества сочетаний двух чисел по восемь) различных комбинаций 0 и 1: от 00000000 до 11111111..

Но этим представлением, называемым двоичным (двоичная система счисления), тяжело пользоваться, хотя в ряде случаев оно очень наглядно. Поэтому более охотно применяют десятичное (десятичная система счисления) или шестнадцатеричное представление (шестнадцатеричная система счисления).

Десятичное представление

Десятичное представление, очевидно, является более естественным, поскольку мы используем десятичную систему счисления. При этом

каждый байт представляется целым числом, находящимся в интервале между 0 и 255.

Переход от одной системы счисления к другой осуществляется представлением каждого бита десятичным числом – «весом» – в зависимости от его ранга (положения) в байте: 1, 2, 4, 8, 16, 32, 64, 128 (восемь первых целых степеней 2, начиная с нулевой степени).

Как правило, бит с наименьшим весом (или младший значащий разряд) расположен справа, а бит с наибольшим весом (или старший значащий разряд) – слева, но иногда бывает удобно использовать обратное расположение разрядов в байте.

При таких условиях двоичное слово 11001011 эквивалентно десятичному числу 128 + 64 + 8 + 2 + 1, или 203.

Шестнадцатеричное представление

«Знатоки» предпочитают в большинстве случаев оперировать шестнадцатеричной системой счисления, то есть системой с основанием шестнадцать. Каждый байт разделяется на две группы по четыре бита, называемые полубайтами, или нибблами (nibble), которые могут принимать шестнадцать различных значений (два в четвертой степени или количество сочетаний двух чисел по четыре).

Эти шестнадцать величин (0000 – 1111) представляются в виде одного символа: цифрами от 0 до 9 или буквами от A до F, согласно правилам, приведенным в табл. 2.1.

Следовательно, целый байт представляется посредством двух символов: это старший значащий полубайт (большего веса) и следующий за ним младший значащий полубайт (меньшего веса).

Если мы воспользуемся нашим предыдущим примером, то байт 11001011 (203 в десятичной системе счисления) можно разделить на 1100 и 1011 или соответственно на C и B. Следовательно, запишем CB в шестнадцатеричной системе счисления.

Разумеется, этот принцип может также применяться для слов, состоящих большее чем из восьми бит. Слово из шестнадцати бит, например, можно разделить на четыре полубайта; таким образом, оно будет представлено четырьмя шестнадцатеричными символами (от 0000 до FFFF).

Чтобы избежать возможной путаницы, к числу часто добавляют еще один знак, предназначенный для уточнения основания используемой системы счисления: например, CBh (шестнадцатеричная) или 203d (десятичная).

Таблица 2 1. Двоичное представление шестнадцатеричных чисел

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Взаимопреобразования между десятичным и шестнадцатеричным представлениями величины байтов удобно выполнять с помощью табл. 2.2 – таблицы соответствия, которая объединяет все 256 возможных случаев.

Представление ASCII

В компьютерах байты часто используют для представления символов клавиатуры, экрана или принтера (буквы, цифры, различные знаки). В табл. 2.3 воспроизведена таблица ASCII (American Standart Code For Information Interchange – американский стандартный код для обмена информацией, «аски»), где устанавливаются эти соответствия.

Определено соответствие только первых 128 байт, заключенных между 0 и 127, так как для следующих 128 существуют многочисленные варианты (речь идет в большинстве случаев о псевдографических символах или о символах других языков). Отметим, впрочем, что и первые 32 кода соответствуют специальным функциям. Например, байт 7 не отображает символ на экране, но приводит к подаче звукового сигнала (bell в переводе с англ. – колокол, в данном случае – колокольчик или звонок). Эти непечатаемые символы называются управляющими кодами.

Таблица 2.2. Десятичное представление шестнадцатеричных чисел

	2-я цифра															
1-я цифра	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	43	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Таблица 2.3 Соответствие между десятичной, шестнадцатеричной системами счисления и ASCII

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
0	00H	NUL	32	20H	SP	64	40H	@	86	60H	'
1	01H	SOH	33	21H	!	65	41H	A	97	61H	A
2	02H	STX	34	22H	"	66	42H	B	98	62H	B
3	03H	ETX	35	23H	#	67	43H	C	99	63H	C
4	04H	EOT	36	24H	$	68	44H	B	100	64H	D
5	05H	ENQ	37	25H	%	69	45H	E	101	B5H	E
6	06H	ACK	38	26H	&	70	46H	F	102	66H	F
7	07H	BEL	39	27H	,	71	47H	G	103	67H	G
8	08H	BS	40	28H	(	72	48H	H	104	68H	H
9	09H	HT	41	29H	)	73	49H	I	105	69H	I
10	0AH	LF	42	2AH	*	74	4AH	J	106	6AH	J
11	0BH	VT	43	2BH	+	75	4BH	K	107	6BH	K
12	0CH	FF	44	2CH	,	76	4CH	L	108	6CH	L
13	0DH	CR	45	2DH	-	77	4DH	H	109	6DH	M
14	0EH	SO	46	2EH	.	78	4EH	N	110	6EH	N
15	0FH	SI	47	2FH	/	79	4FH	0	111	6FH	0
16	10H	DLE	48	30H	0	80	50H	P	112	70H	P
17	11H	DC1	49	31H	1	81	51H	Q	113	71H	Q
18	12H	DC2	50	32H	2	82	52H	R	114	72H	R
19	13H	DC3	51	33H	3	83	53H	S	115	73H	S
20	14H	DC4	52	34H	4	84	54H	T	116	74H	T
21	15H	NAK	53	35H	5	85	56H	U	117	75H	U
22	16H	SYN	54	36H	6	86	57H	V	118	76H	V
23	17H	ETB	55	37H	7	87	58H	W	119	77H	W
24	18H	CAN	56	38H	8	88	58H	X	120	78H	X
25	19H	EM	57	39H	9	89	59H	Y	121	79H	Y
26	1AH	SUB	58	3AH	:	90	5AH	Z	122	7AH	Z
27	IBH	ESC	59	3BH	;	91	5BH	[	123	7BH	{
28	1CH	FS	60	3CH	<	92	5CH	\	124	7CH	|
29	1DH	GS	61	3DH	=	93	5DH	]	125	7DH	}
28	1CH	FS	60	3CH	<	92	5CH	\	124	7CH	|
29	1DH	GS	61	3DH	=	93	5DH	]	125	7DH	}
30	1EH	RS	62	3EH	>	94	5EH	^	126	7EH	~
31	1FH	US	63	3FH	?	95	5FH	_	127	7FH	DEL

ФАЙЛЫ

Содержимое запоминающего устройства, которое должно быть обработано на компьютере или просто сохранено на дискете, называют *файлом*.

С точки зрения информатики термин «файл» обозначает последовательность байтов, размещенных на каком-либо носителе, которую можно легко защитить, копировать, перемещать или изменять с помощью обычного программного обеспечения.

Двоичные файлы

Очевидно, что записать байты такими, какими они представлены в запоминающем устройстве, наиболее просто. Такой файл называют бинарным, или двоичным.

Для примера рассмотрим последовательность из 22 десятичных значений, представленных ниже в виде обычного текста:

```
194 128 175 255 174 255 222 254
223 250 210 128 175 255 174 255
222 254 223 250 128 234
```

Если мы запишем их в файл в виде 22 соответствующих байтов, листинг на экране или на принтере получится таким, как он показан ниже (по крайней мере, на ЭВМ, совместимой с ПК). Пример двоичного файла представлен в символах ASCII.

```
┬ç» « ▐▀■·┬ç» « ▐▀■·çΩ
```

Возможно, это самая компактная по количеству символов форма записи байтов, но и самая непонятная.

Десятичный файл

Более удобной формой записи является *свободный десятичный формат* (free format decimal). Чтобы сохранить данные в этом формате, байты надо записывать соответствующими десятичными числами, разделенными пробелами. Таким образом, до четырех байт текста в файле соответствуют одному байту в запоминающем устройстве. Взамен – наглядность изменений, которые легко внести даже с помощью простейших программ, находящихся в распоряжении любого пользователя ПК.

Шестнадцатеричный файл

Тот же самый принцип может быть применен и к шестнадцатеричным величинам, требующим только двух символов для представления байта информации. Если, кроме того, исключить разделительные

пробелы, получим представление, известное под названием *сплошной гекс* (straight hex), то есть только шестнадцатеричные. Пример файла в формате straight hex:

```
C280AFFFAEFFDEFEDFFAD280AFFFAEFFDEFEDFFA80EA
```

Занимающий только два байта текста на один байт в запоминающем устройстве (и, возможно, несколько специальных символов в конце строки), формат straight hex – это хороший компромисс между компактностью и наглядностью.

Однако разработаны и довольно широко используются улучшенные варианты форматов шестнадцатеричных файлов, из которых наиболее известны Intel Hex и Motorola S (названия соответствуют именам компаний-производителей радиоэлементов, разработавших эти форматы).

Главное их преимущество – возможность включать специальные контрольные ключи, позволяющие обнаруживать ошибки, возникающие во время передачи или копирования файлов, а также наличие информации, определяющей, для каких адресов или области адресов запоминающего устройства предназначены байты. Уменьшение размера файла может быть очень существенным в случае, если запоминающее устройство заполнено только частично.

Файл Intel Hex

Ниже приведен пример в формате файла Intel Hex:

```
:10000000C280AFFFAEFFDEFEDFFAD280AFFFAEFFF1
·06001000DEFEDFFA80EACB
:00000001FF.
```

Каждая строка файла, или *запись*, начинается с девяти символов, которые имеют следующие стандартные значения:

символ 1	символ «двоеточие»;
символы 2 и 3	количество значащих байтов в записи (здесь 16, или 10h в шестнадцатеричном коде);
символы 4 и 5	старший байт адреса размещения этой записи в запоминающем устройстве (в шестнадцатеричном коде);
символы 6 и 7	младший байт адреса (в шестнадцатеричном коде);
символы 8 и 9	не используются (00).

Далее следуют информационные байты, причем каждый состоит из двух символов, как и в формате straight hex, потом – два символа, представляющие байт контрольной суммы (checksum), а затем символы «возврат каретки» и «перевод строки».

Контрольная сумма вычисляется как дополнение до двух суммы восьмиразрядных чисел без учета переноса всех полезных байтов, двух байтов адреса и байта длины строки.

Последняя строка – «признак конца» – не содержит полезных байтов.

Файл Motorola S

Ниже представлены те же данные, но в формате Motorola S:

```
S1130000C280AFFFAEFFDEFEDFFAD280AFFFAEFFED
S1090010DEFEDFFA80EAC7
S9030000FC
```

Здесь каждая строка содержит некоторое количество байтов, организованных следующим образом:

символ 1: символ «S»;

символ 2: определение типа записи:

- 0 – запись заголовка;
- 1 – запись данных при 16-разрядном коде адреса;
- 2 – запись данных при 24-разрядном коде адреса;
- 3 – запись данных при 32-разрядном коде адреса;
- 7 – конец файла при 32-разрядном коде адреса;
- 8 – конец файла при 24-разрядном коде адреса;
- 9 – конец файла при 16-разрядном коде адреса.

Далее следует некоторое количество символов, определяющих размер записи (количество полезных байтов, адрес, контрольный ключ), здесь 19 байт (13h в шестнадцатеричной системе счисления), начальный адрес для размещения данных в запоминающем устройстве (здесь 0000h), байты полезной информации, кодируемые двумя символами, и, наконец, контрольный ключ. Он вычисляется как дополнение до единицы суммы полезных байтов, длины и адреса.

Запись всегда заканчивается символами «возврат каретки» и «перевод строки». Здесь также последняя строка является записью конца и не содержит байтов полезной информации.

Приведенные стандарты важно знать, так как в большинстве имеющихся в продаже программаторов информация должна быть представлена в одном из описанных форматов. Соответственно многие программы и системы электронного проектирования генерируют коды для программирования в том или ином формате.

Достаточно часто может потребоваться перекодировка информации из одного формата в другой для устранения несовместимости оборудования и программного обеспечения разных производителей.

Относительно простые утилиты помогут начинающему разработчику решить и эту задачу (см. главу 6).

ПРИМЕНЕНИЕ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Перечислить все возможные варианты применения запоминающих устройств нельзя, поскольку все определяется комбинациями значений битов, которые в них записаны. Что могут означать 22 байта из нашего примера? Напомним, что любой из них представляется восемью битами, каждый из которых может принимать значения 1 или 0.

Эти 1 или 0 могли бы, например, соответствовать состоянию «замкнуто» или «разомкнуто» для группы из восьми реле. Тогда каждый адрес содержал бы данные для определенного шага программы, лишь бы только специально предназначенный счетчик с соответствующей скоростью по очереди «перебирал» один за другим все адреса.

Эти же 22 байта могли бы представлять выборку длительностью несколько микросекунд из оцифрованной записи звука, которую с помощью простейшего цифро-аналогового преобразователя на сопротивлениях легко воспроизвести через любой усилитель.

В действительности же мы говорим о короткой программе, разработанной в качестве примера для микроконтроллера 8751 и предназначенной для управления свечением сигнальной лампочки, подключенной к одному из выводов микроконтроллера.

На рис. 2.1 приведен исходный текст этой программы, то есть последовательность команд на ассемблере, который может быть переведен непосредственно в исполняемые байты при помощи специальной

```
Line 1      Col 1      Pos 1      Insert        AutoIndent                 A:\CLI
--------------------------------------------------------------------------------
File        blocK      Quick      Error/brEak   Screen      Remember      Options
Delete      plaCe      Ascii      Printer       reBlock     rep Last find     Esc
debut:
 clr p0.0
 mov r7,255
loopa:
 mov r6,255
loopb:
 djnz r6,loopb
 djnz r7,loopa
 setb p0.0
 mov r7,255
loopc:
 mov r6,255
loopd:
 djnz r6,loopd
 djnz r7,loopc
 jmp debut

1 Help  2 Diags 3 FPMs  4 ReAssm5 CurSkp6 Zoom  7 Watch 8 Go     9        0 Config
```

Рис 2.1. Исходный текст программы, соответствующий файлу примера

программы, называемой кросс-ассемблер (она размещена на сайте www.dmk.ru).

Речь идет о простейшем, классическом примере: запрограммированное соответствующим образом запоминающее устройство может выполнять самые разнообразные функции, например заменять стандартный 7-сегментный дешифратор в тех случаях, когда серийные микросхемы не соответствуют требованиям разработчика. А в матричных принтерах это может быть запоминающее устройство, где содержатся изображения всех доступных для печати символов: достаточно перепрограммировать его соответствующим образом, чтобы печатать, например, на греческом языке.

СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Данные, предназначенные для записи в запоминающее устройство, можно, конечно, сначала представить на бумаге, а затем загрузить вручную с помощью простейшего программатора. Действительно, это вполне осуществимо и допустимо при создании устройств, для которых необходимо запрограммировать от нескольких десятков до нескольких сотен байтов.

Но счастливые владельцы микро-ЭВМ, особенно полностью совместимых с IBM PC, все еще не подозревают о скрытом в компьютере могуществе и тех преимуществах, которые он дает при разработке и отладке аппаратуры или систем, где используются различные запоминающие устройства.

Некоторые стандартные утилиты из состава операционных систем уже могут сослужить добрую службу, когда понадобится просто вывести на экран в удобном виде двоичные файлы или изменить их форматы.

На рис. 2.2 в качестве примера представлено изображение экрана монитора при работе с программой PC-TOOLS, используемой для работы с двоичным файлом, приведенным на стр. 21, где отображаются 22 шестнадцатеричных кода (см. наш пример в представлении straight hex на стр. 21), за которыми следует бессмысленный набор символов (байтов), поскольку эта утилита работает с файлом, размещенным на дискете.

Пример, представленный на рис. 2.3, иллюстрирует очень экономичный способ приобрести превосходное программное обеспечение в виде демонстрационной дискеты, часто поставляемой как бесплатное приложение к покупному программатору.

Часто с помощью таких демонстрационных версий можно обрабатывать различные файлы и даже перекодировать их в те или иные основные форматы, которые мы описали выше.

```
PC Tools Deluxe R4.30                                         Label Vol=Aucun
-------------------Service Visualisation/Modification Fichier------------------
Voie=A:\*.*
Fichier=CLI.BIN    Secteur Relatif 0000000 Groupe 00680 Sect abs disque 0001370

D placement  ----------------- Codes hex--------------------    Valeur ASCII
 0000(0000)  C2 80 AF FF AE FF DE FE DF FA D2 80 AF FF AE FF    +
 0016(0010)  DE FE DF FA 80 EA 8C 8D E0 AE 9B 8D 9B 8E 8A 9B
 0032(0020)  9C D2 E0 AA 9B 8E 8D 97 91 92 E0 CE D2 CF CA E0
 0048(0030)  AA 9B 8E 8D 97 91 92 E0 CE D2 CF CA E0 AA 9B 8E
 0064(0040)  D0 08 01 0B 52 37 39 37 3E 00 D2 DE B7 AD C8 CA        R797>      =
 0080(0050)  CA 19 FA F2 1C 2B 58 13 08 AE E1 FC F4 FA 4E 3D     v    +X .      N=
 0096(0060)  3E 3E 31 59 48 D5 E8 31 31 BA 06 01 0F 08 91 9C    >>1YH  11"    .
 0112(0070)  5D 3D A1 86 DF 3B 14 83 93 1A BE 58 80 8D 1D 83    ]=    ;      X
 0128(0080)  DE 95 67 85 0C 85 F4 81 CE 81 E8 90 53 89 6F 74      g           S ot
 0144(0090)  FE 87 FD 98 3C A0 48 A0 08 96 AF 9F C3 95 28 97        < H .    + (
 0160(00A0)  DA 2D 1B 1D 0B 0E 17 0C 07 53 35 1B 07 60 33 17    +-          S5  .3
 0176(00B0)  0D 0D 17 12 19 52 52 52 52 52 52 3B 3D 3F 34 60        vRRRRRR;=?4.
 0192(00C0)  82 8D 81 87 7E 84 83 13 E9 34 A5 7F 2B 1A DA E7        ~      4 .+ +
 0208(00D0)  E2 E6 30 62 4C 61 BE 5D 6C 21 82 DD 1D 6A AF 33      ObLa ]l!    j 3
 0224(00E0)  2A EE A1 FE 08 7D 7B 03 20 20 20 20 00 00 00 00    *    .}{
 0240(00F0)  00 00 00 00 00 00 BA 4D E2 14 5D 14 C5 0D 00 00           "M   ]

 Home=début de fichier/disque End=fin de fichier/disque
 ESC=Sortie  PgDn=avant  PgUp=arrière  F1=ASCII/HEX  F2=chg n  secteur  F3=modif
```

Рис. 2.2 Файл в программе PC-TOOLS

```
                  ----------------------- Bytes -----------------------
Buffer Co | Addr      Hex            Binary
Clear Edi |         0: C2 80 AF FF   11000010 10000000 10101111 11111111
          |         4: AE FF DE FE   10101110 11111111 11011110 11111110
          |         8: DF FA D2 80   11011111 11111010 11010010 10000000
          |         C: AF FF AE FF   10101111 11111111 10101110 11111111
a         |        10: DE FE DF FA   11011110 11111110 11011111 11111010
          |        14: 80 EA 00 00   10000000 11101010 00000000 00000000
          |        18: 00 00 00 00   00000000 00000000 00000000 00000000
          |        1C: 00 00 00 00   00000000 00000000 00000000 00000000
          |        20: 00 00 00 00   00000000 00000000 00000000 00000000
          |        24: 00 00 00 00   00000000 00000000 00000000 00000000
          |        28: 00 00 00 00   00000000 00000000 00000000 00000000
          |        2C: 00 00 00 00   00000000 00000000 00000000 00000000
          |        30: 00 00 00 00   00000000 00000000 00000000 00000000
          |        34: 00 00 00 00   00000000 00000000 00000000 00000000
          |        38: 00 00 00 00   00000000 00000000 00000000 00000000
          |        3C: 00 00 00 00   00000000 00000000 00000000 00000000
          |        40: 00 00 00 00   00000000 00000000 00000000 00000000
          |        44: 00 00 00 00   00000000 00000000 00000000 00000000
          |        48: 00 00 00 00   00000000 00000000 00000000 00000000
Buffer:  C|        4C: 00 00 00 00   00000000 00000000 00000000 00000000  um: 121EH
Device:  N|             CURSOR:           0 END:        15
Options: L
   F1helpF2radixF3gotoF4reconfigureF5searchF6fillF7copyF8invertF9checksumF10exit
```

Рис. 2.3. Пример редактора файлов «зашивок» для ППЗУ

Подобные программы – превосходная форма рекламы, ведь, используя это программное обеспечение для самодельного программатора, покупатель при приобретении более совершенного и производительного прибора будет отталкиваться от того, что ему наиболее знакомо, с чего он начинал.

Среди всех демонстрационных версий, которые мы сумели получить, особенно интересно программное обеспечение программаторов

производства компаний MQP, BP Microsystems и Sprint. Образцы этих программ представлены на сайте www.dmk.ru.

Наш пример показывает, как редактор файлов компании BP Microsystems позволяет представлять и редактировать файл в двоичном или шестнадцатеричном формате (а также ASCII), каков бы ни был формат, в котором он сохранен.

Для получения машинного кода микропроцессора специальная программа-ассемблер обрабатывает те исходные тексты, размер которых превышает несколько десятков команд, поскольку короткие программы вполне можно ассемблировать вручную.

Демонстрационная версия универсального кросс-ассемблера ECAL позволяет на обычном ПК сгенерировать код практически для любого микропроцессора или микроконтроллера. Возможности данной версии ограничены: например, нельзя генерировать код длиной более 256 байт, но этого достаточно для небольших первых проектов и изделий, а в дальнейшем можно комбинировать несколько подготовленных раздельно коротких программ в одном и том же запоминающем устройстве.

Кроме того, очень полезной может стать программа-дизассемблер, служащая для восстановления исходного текста программы из исполняемого кода, размещенного в запоминающем устройстве. На рис. 2.4 приведен результат работы дизассемблера, с помощью которого был проанализирован наш файл, представленный в символах ASCII.

```
-------------------- AMS 8032 TARGET PROGRAM EDIT SCREEN --------------------
field up       char left      Ctrl=field left    Pgup=prev page
 ESC =EXIT     field down     char right   Ctrlfield right   Pgdn=next page :
 F2  =tree   F3=index         F6=symbolics[OFF]  F7=instructions    F8=list[OFF]
                                 Symbolic entry- type '.' then symbolic name

0000    C280                         CLR      80
0002    AFFF                         MOV      R7,FF
0004    AEFF                         MOV      R6,FF
0006    DEFE                         DJNZ     R6,0006
0008    DFFA                         DJNZ     R7,0004
000A    D280                         SETB     80
000C    AFFF                         MOV      R7,FF
000E    AEFF                         MOV      R6,FF
0010    DEFE                         DJNZ     R6,0010
0012    DFFA                         DJNZ     R7,000E
0014    80EA                         SJMP     0000
0016    00                           NOP
0017    00                           NOP
0018    00                           NOP
0019    00                           NOP
001A    00                           NOP
001B    00                           NOP
001C    00                           NOP
PC    MACHINE    LABEL      OPCD     OPERAND OPERAND
```

Рис. 2.4. Файл, обработанный дизассемблером

Разумеется, интересное программное обеспечение частного применения может быть написано на языках Basic или Pascal, которые широко распространены среди пользователей ПК. Много полезных действий можно выполнить с помощью программ из нескольких десятков строк.

ПРОГРАММИРУЕМЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Если мы решили считать программируемыми радиоэлементами только те, которые с помощью относительно простых устройств могут быть настроены окончательным образом или, по крайней мере, на длительный срок, тогда лишь несколько семейств запоминающих устройств заслуживают названия по-настоящему программируемых.

Динамические или обыкновенные статические ОЗУ должны быть сразу же исключены, поскольку они теряют хранящуюся в них информацию после отключения питания.

ОЗУ со встроенным элементом питания и ЭСППЗУ могут рассматриваться как пограничный случай, так как, с одной стороны, они могут сохранять записанные в них данные после снятия напряжения питания в течение нескольких лет, а с другой стороны – всегда остается возможность перезаписать в них новые данные, как в обычное ОЗУ. Таким образом, их содержание зафиксировано не вполне надежно, поскольку есть вероятность случайного изменения их содержимого.

Масочные ПЗУ, данные в которые заносятся в процессе изготовления на заводе, не могут быть изменены пользователем. Поэтому их следует отнести к категории «заказных» микросхем.

ППЗУ, то есть ПЗУ, которые программируются за счет необратимого разрушения внутренних плавких перемычек, можно делать как в единичном экземпляре, так и партией с помощью соответствующих программаторов. Таким образом, ППЗУ, несомненно, являются программируемыми радиоэлементами. Главное их преимущество – надежность записанной информации: нужно затратить значительное количество энергии, чтобы расплавить перемычку, восстановить же ее невозможно. (В общем случае это не совсем так: в ППЗУ низкого качества вполне вероятно восстановление определенной части перемычек за счет кристаллизации материала перемычки на ее остатках – вплоть до нормального соединения – из паров, скапливающихся во внутренней полости корпуса микросхемы. Но этот процесс занимает несколько лет, и поврежденные таким образом биты можно вновь перепрограммировать.)

Такие радиодетали невозможно основательно протестировать на заводе, поскольку для этого пришлось бы уничтожить все плавкие перемычки. Считается нормальным, что для некоторого процента микросхем ППЗУ программирование будет неудачным, поэтому необходима полная проверка содержимого каждой микросхемы после ее программирования.

СППЗУ являются ПЗУ, программируемыми пользователем, и стираются простым облучением ультрафиолетовыми лучами. Если микросхемы выпускаются в керамических корпусах с окошком из кварцевого стекла (материала, прозрачного для ультрафиолета), они перепрограммируются несколько сотен раз.

Если же кристаллы таких микросхем размещены в непроницаемых для УФ излучения пластмассовых корпусах, информация из них не может быть стерта, и тогда они называются ОП (однократно программируемые). Цена на них соперничает с ценой на ППЗУ с плавкими перемычками, но программирование СППЗУ требует гораздо меньших затрат энергии. К тому же с них легко стирать информацию во время тестирования на заводе еще до корпусирования, поэтому перед поставкой они стопроцентно тестируются.

Хранящаяся здесь информация, однако, должна рассматриваться как менее стабильная по сравнению с ППЗУ с плавкими перемычками. Действительно, обычно в СППЗУ с течением времени идет процесс очень медленного стирания информации под влиянием утечек внутри кристалла, температуры, повышение которой ведет к росту утечек, а также за счет различных ионизирующих излучений. Но считается, что в обычных условиях не следует ничего опасаться по крайней мере в течение десяти лет.

Таким образом, СППЗУ – лучший выбор для среднего пользователя. Эти радиоэлементы более популярны, чем ППЗУ с плавкими перемычками или запоминающие устройства типа флэш (так и не сумевшие вытеснить СППЗУ). Благодаря постоянному улучшению технологии производства и методов программирования СППЗУ становятся все более конкурентоспособными, а кроме того, в некоторых новых типах используются различные полезные нововведения.

Основные цоколевки микросхем ПЗУ

Хотя технология СППЗУ позволяет реализовать запоминающие устройства практически любой архитектуры, наиболее распространенной все еще остается организация из слов по восемь бит. В СППЗУ, статических запоминающих устройствах, обычно применяют стандарт

Bytewide (однобайтный, длиной в один байт), который предназначен для цоколевок максимального количества типов статических запоминающих устройств, организованных в восьмиразрядные слова и относящихся к следующим типам: ОЗУ, ПЗУ, ППЗУ, ЭСППЗУ, СППЗУ, флэш и т.д.

Эта стандартизация позволяет, с некоторыми ограничениями, использовать на разных этапах в одной и той же панельке запоминающие устройства стандарта Bytewide различных семейств и разного объема, что оказывается на практике весьма полезным.

Ниже представлена табл. 2.4, объединяющая цоколевки наиболее распространенных УФ СППЗУ, выполненных в корпусе DIP с 24 выводами.

Очевидно, что выводы «общий» и V_{CC} (питание +5 В) для различных моделей одни и те же. Речь идет соответственно о выводах 12 и 24, расположенных по диагонали корпуса, как и в случае большого числа современных стандартных логических схем.

Восемь линий данных (D0 – D7) также занимают стандартизованные места: выводы 9–11 и 13–17. Именно через них осуществляется параллельный обмен данными, то есть сразу по восьми линиям (по 8 бит), когда запоминающее устройство программируется или из него считывается информация.

2716	2732 27C32	27C16	DIP 24 выв.		27C16	2732 27C32	2716
A_7	A_7	A_7	1	24	V_{CC}	V_{CC}	V_{CC}
A_6	A_6	A_6	2	23	A_8	A_8	A_8
A_5	A_5	A_5	3	22	A_9	A_9	A_9
A_4	A_4	A_4	4	21	V_{PP}	A_{11}	V_{PP}
A_3	A_3	A_3	5	20	$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$
A_2	A_2	A_2	6	19	A_{10}	A_{10}	A_{10}
A_1	A_1	A_1	7	18	$\overline{CE}/\overline{PGM}$	$\overline{CE}$	$\overline{CE}/PGM$
A_0	A_0	A_0	8	17	D_7	D_7	D_7
D_0	D_0	D_0	9	16	D_6	D_6	D_6
D_1	D_1	D_1	10	15	D_5	D_5	D_5
D_2	D_2	D_2	11	14	D_4	D_4	D_4
⏚	⏚	⏚	12	13	D_3	D_3	D_3

Рис. 2.5. Цоколевки основных УФ СППЗУ с 24 выводами

Таблица 2.4. Архитектура некоторых СППЗУ

Тип	Кбит	Кб	Адресные линии	N = число адресных линий	2^N
2716	16	2	A0 – A10	11	2048
2732	32	4	A0 – A11	12	4096
2764	64	8	A0 – A12	13	8192
27128	128	16	A0 – A13	14	16384
27256	256	32	A0 – A14	15	32768
27512	512	64	A0 – A15	16	65536

Количество линий адреса зависит у разных микросхем от объема памяти конкретного запоминающего устройства. Рис. 2.6 поясняет, как N адресных линий определяют 2^N адреса, или «ячейки памяти», либо 8×2^N бит, так как по каждому адресу может храниться байт, содержащий 8 бит.

Поскольку в цифровой технике используются, как правило, степени числа два, стало уже привычным понимать под приставкой «кило» число 1024 вместо 1000 (1024 – десятая степень числа 2).

Поэтому запоминающее устройство типа 2716 на 16 Кбит содержит 2 Кб либо 2048×8 бит, иначе говоря – 16384 бита.

Каждый раз, когда объем запоминающего устройства удваивается, добавляется одна линия адреса. Применение стандарта Bytewide выражается в том, что назначение выводов микросхем разной емкости одинаково. Для дополнительной адресной линии выделяется вывод, не использовавшийся ранее, или освобождается одна из служебных линий, объединяя две функции на одном выводе.

Так, у типа 2716 применяется два различных вывода для подачи напряжения программирования V_{PP} и для сигнала разрешения чтения $\overline{OE}$. Но эти два сигнала могут очень хорошо передаваться по одному и тому же выводу: напряжение V_{PP} необходимо только при программировании, тогда как сигнал $\overline{OE}$ может быть в низком логическом уровне только в режиме чтения.

Таким образом, у микросхем 2732 на выводе 20 комбинируются сигналы V_{PP} и $\overline{OE}$, и вывод 21 освобождается для дополнительной адресной линии A11.

На рис. 2.7 показано, как этот принцип сохраняется у микросхем с объемом памяти больше 4 Кб за счет использования корпуса с 28 выводами. Таким образом, можно достичь объема памяти в 64 Кб в микросхемах 27512, полностью поддерживая совместимость со стандартом Bytewide на уровне 24 нижних (по рисунку) выводов корпуса

27512 27C512	27256 27C256	27128 27C128	2764 27C64	DIP 24 выв.		2764 27C64	27128 27C128	27256 27C256	27512 27C512
A_{15}	V_{PP}	V_{PP}	V_{PP}	1	28	V_{CC}	V_{CC}	V_{CC}	V_{CC}
A_{12}	A_{12}	A_{12}	A_{12}	2	27	$\overline{PGM}$	$\overline{PGM}$	A_{14}	A_{14}
A_7	A_7	A_7	A_7	3	26	Н.С.	A_{13}	A_{13}	A_{13}
A_6	A_6	A_6	A_6	4	25	A_8	A_8	A_8	A_8
A_5	A_5	A_5	A_5	5	24	A_9	A_9	A_9	A_9
A_4	A_4	A_4	A_4	6	23	A_{11}	A_{11}	A_{11}	A_{11}
A_3	A_3	A_3	A_3	7	22	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$
A_2	A_2	A_2	A_2	8	21	A_{10}	A_{10}	A_{10}	A_{10}
A_1	A_1	A_1	A_1	9	20	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
A_0	A_0	A_0	A_0	10	19	D_7	D_7	D_7	D_7
D_0	D_0	D_0	D_0	11	18	D_6	D_6	D_6	D_6
D_1	D_1	D_1	D_1	12	17	D_5	D_5	D_5	D_5
D_2	D_2	D_2	D_2	13	16	D_4	D_4	D_4	D_4
⏚	⏚	⏚	⏚	14	15	D_3	D_3	D_3	D_3

Рис. 2.6. Цоколевка основных типов СППЗУ в корпусе с 28 выводами

или панельки. Это означает, что в хорошо продуманной схеме (программаторе или разрабатываемом изделии) допустимо применять СППЗУ самой разной емкости. В худшем случае придется переставить несколько заранее предусмотренных перемычек.

В начале 1990-х годов наиболее распространенным типом СППЗУ была микросхема 2764 или ее КМОП версия 27C64, которая стоила дешевле, чем 2716, несмотря на увеличенный в четыре раза объем памяти. А еще три или четыре года назад мы со всей уверенностью писали, что 2716 является наиболее экономически выгодным и рекомендуемым СППЗУ.

Сегодня панельки с 28 выводами постепенно вытесняют панельки с 24 выводами, и ничто не мешает запрограммировать только четверть объема микросхемы 27C64 данными, предназначенными для 2716: для этого достаточно лишь выставить низкий уровень на двух избыточных адресных линиях (A11 и A12).

Разумеется, существуют СППЗУ с меньшим объемом памяти, чем у 2716, а теперь встречаются СППЗУ емкостью в несколько мегабит.

Тем не менее множество вещей можно делать уже на микросхемах 27С256 и 27С512, которые сейчас являются наиболее доступными из СППЗУ, имеющихся в розничной торговле.

На рис. 2.7 приведена цоколевка микросхемы 2708 (1 Кб), которая даже для чтения требовала дополнительных напряжений, превышающих 5 В.

На рис. 2.8 показано, каким образом у микросхемы 27С010 достигается объем памяти в 1 Мбит (1024 Кбит) за счет использования четырех дополнительных выводов (корпус DIP с 32 выводами), но при соблюдении совместимости по стандарту Bytewide с моделями, имеющими 28 или 24 вывода.

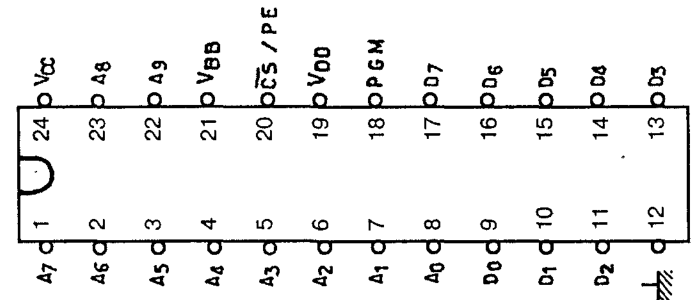


Рис. 2.7. Цоколевка микросхемы 2708 (1 Кб)

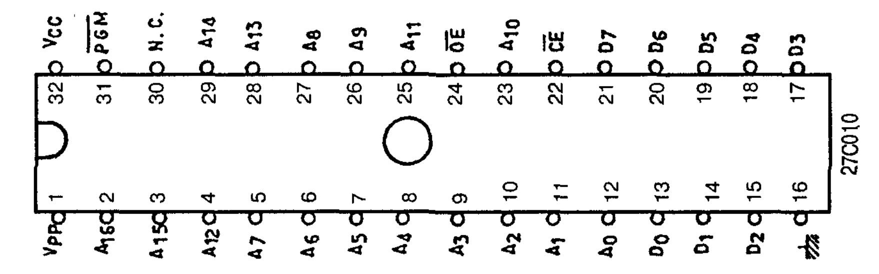


Рис. 2.8. Цоколевка микросхемы 27С010 (1 Мбит)

Алгоритмы программирования

Алгоритмом программирования называется последовательность операций, необходимых для правильного программирования: какие сигналы подавать на определенные выводы, какое напряжение и в течение какого времени удерживать, как контролировать успешность завершения каждого этапа, как проверить работоспособность готового программируемого радиоэлемента и т.д.

Технология СППЗУ предполагает, что программирование происходит под напряжением гораздо выше, чем 5 В, которое применяется в режиме чтения. Прежде рекомендовавшееся для всех типов СППЗУ напряжение V_{PP} равное 25 В уменьшилось до 21 В, а затем и до 12,5 В. Надлежит точно соблюдать величину, предписанную изготовителем (иногда она маркируется на корпусе микросхемы, но не всегда), иначе можно существенно сократить срок службы СППЗУ.

Напряжение V_{PP}, как правило, подается на специальный вывод, а V_{CC} всегда должно быть равно 5 В или немного больше. (Недавно появилась тенденция увеличивать напряжение V_{CC} до 6 В во время программирования для лучшего «запоминания» данных.) Отметим, что V_{PP} обязательно должно быть приложено одновременно или после подачи V_{CC} и выключить его необходимо одновременно или до отключения V_{CC}.

После включения V_{CC} и V_{PP} на микросхему подается байт для программирования по восьми линиям данных, одновременно с этим на линиях адреса выставляются логические уровни, определяющие номер ячейки, в которую должны быть записаны данные.

Как правило, программируются все адреса в порядке их возрастания, но можно начать с любого заранее выбранного адреса.

Калиброванный импульс (почти всегда отрицательный, но в особом случае с 2716 положительный) подается после этого на специальный вывод ($\overline{PGM}$). Раньше длительность этого импульса была одинаковой, равной 50 мс, а длительность программирования составляла для микросхемы 2716 около двух минут и тогда считалась нормальной.

Легко подсчитать, что при этих условиях на программирование микросхемы 27512 потребовался бы почти час, что, очевидно, совершенно неприемлемо. Таким образом, для новых СППЗУ достаточно импульса длительностью 10 мс или меньше. Кроме того, существуют различные способы (быстрые алгоритмы), позволяющие ускорить процесс программирования.

В табл. 2.5 указана минимальная продолжительность процесса программирования, характерная для основных типов микросхем в случае программирующих импульсов длительностью 50, 10 и 1 мс.

Для контрольного чтения, необходимого при проверке правильности проведенного программирования, требуется обычно одна-две минуты. Запоминающее устройство переводится в обычный режим чтения, в частности подавая низкий уровень на вывод $\overline{CE}$ (выбор кристалла) и на $\overline{OE}$ (разрешение чтения).

Таблица 2.5. Длительность процесса программирования

	2716	2732	2764	27128	27256	27512
1 мс	2 с	4 с	8 с	16 с	32 с	1 мин 4 с
10 мс	20 с	40 с	1 мин 20 с	2 мин 40 с	5 мин 20 с	10 мин 40 с
50 мс	1 мин 42 с	3 мин 25 с	6 мин 50 с	13 мин 40 с	27 мин 18 с	54 мин 37 с

Следует отметить, что количество алгоритмов программирования почти равно числу производителей запоминающих устройств по всему миру, и даже больше: их почти столько же, сколько выпускается разновидностей микросхем СППЗУ. Нередко микросхема 2764 марки X программируется при напряжении 25 В импульсом программирования 50 мс (весьма архаичный способ), а для другой, марки Y, достаточно нескольких миллисекунд при напряжении 12,5 В.

Прежде чем пытаться программировать СППЗУ, не пренебрегайте чтением технического описания, справочного листка или обращением к какой-нибудь информационной базе данных, чтобы выяснить особенности эксплуатации соответствующей марки.

Если у вас возникли сомнения (например, марка стерта), можно начать с «мягких» режимов (допустим, 12,5 В и 10 мс) и сразу же выполнить контрольное считывание. В случае неудачи нужно постепенно увеличивать напряжение и/или длительность импульса (но только в случае неудачи).

Быстрые алгоритмы

Даже с программирующими импульсами продолжительностью 10 мс для 27C512 потребуется более 10 мин. Поэтому производители СППЗУ предлагают различные более или менее сложные процедуры программирования, позволяющие экономить время.

Одна из наиболее очевидных процедур, которую возможно применять практически всегда, состоит в том, чтобы сравнивать значение программируемого байта с содержанием ячейки, куда производится запись. Если имеет место равенство, то незамедлительно переходят к следующему адресу, выигрывая таким образом до 50 ценных миллисекунд. В противном случае, конечно, выполняют обычное программирование по данному адресу. Такая процедура может оказаться достаточно рентабельной, потому что СППЗУ (главным образом большого объема) нередко заполнены только частично.

Другая хитрость заключается в использовании для программирования очень коротких импульсов (0,1–1 мс), причем процесс повторяется столько раз, сколько потребуется для успешного завершения операции. Таким образом, некоторые алгоритмы построены по принципу «ровно столько, сколько надо», в то время как в других используются несколько дополнительных программирующих импульсов.

Кроме экономии времени, данные процедуры дают возможность максимально использовать СППЗУ. Однако это предполагает применение достаточно сложных программаторов, в которых можно непрерывно чередовать циклы программирования и контрольного считывания.

Количество попыток программирования должно быть ограничено: в случае неудачи по завершении определенного времени действия импульсов запоминающее устройство объявляется неисправным.

На рис. 2.9 приведен алгоритм, рекомендуемый компанией NS (National Semiconductor) для ряда своих КМОП СППЗУ. Он соответствует максимальной длительности импульса 10 мс, который можно применять, если программатор не приспособлен к быстрым алгоритмам.

На рис. 2.10 показан алгоритм, с помощью которого компания Atmel рекомендует применять «допрограммирование» импульсом 3 мс для закрепления успешной операции. Таким образом, время программирования одной ячейки находится в интервале между 4 и 25 мс.

В компании Cypress (рис. 2.11) используют очень короткие импульсы (100 мкс). Как только обрабатываемая ячейка представляется запрограммированной, применяют дополнительный импульс, длительность которого изменяется в зависимости от того, сколько попыток записи в эту ячейку уже было предпринято (так называемый адаптивный алгоритм).

Для максимальной надежности каждый производитель рекомендует применять именно те алгоритмы, которые он предлагает и которые, по его словам, оптимизированы в зависимости от технологии, используемой для изготовления микросхем СППЗУ.

В повседневной работе можно, однако, позволить себе несколько свободнее подходить к таким процедурам. При этом появляется возможность использовать упрощенный программатор, более дешевый, чем стандартные модели. Но в таком случае придется отказаться от гарантий производителя микросхем.

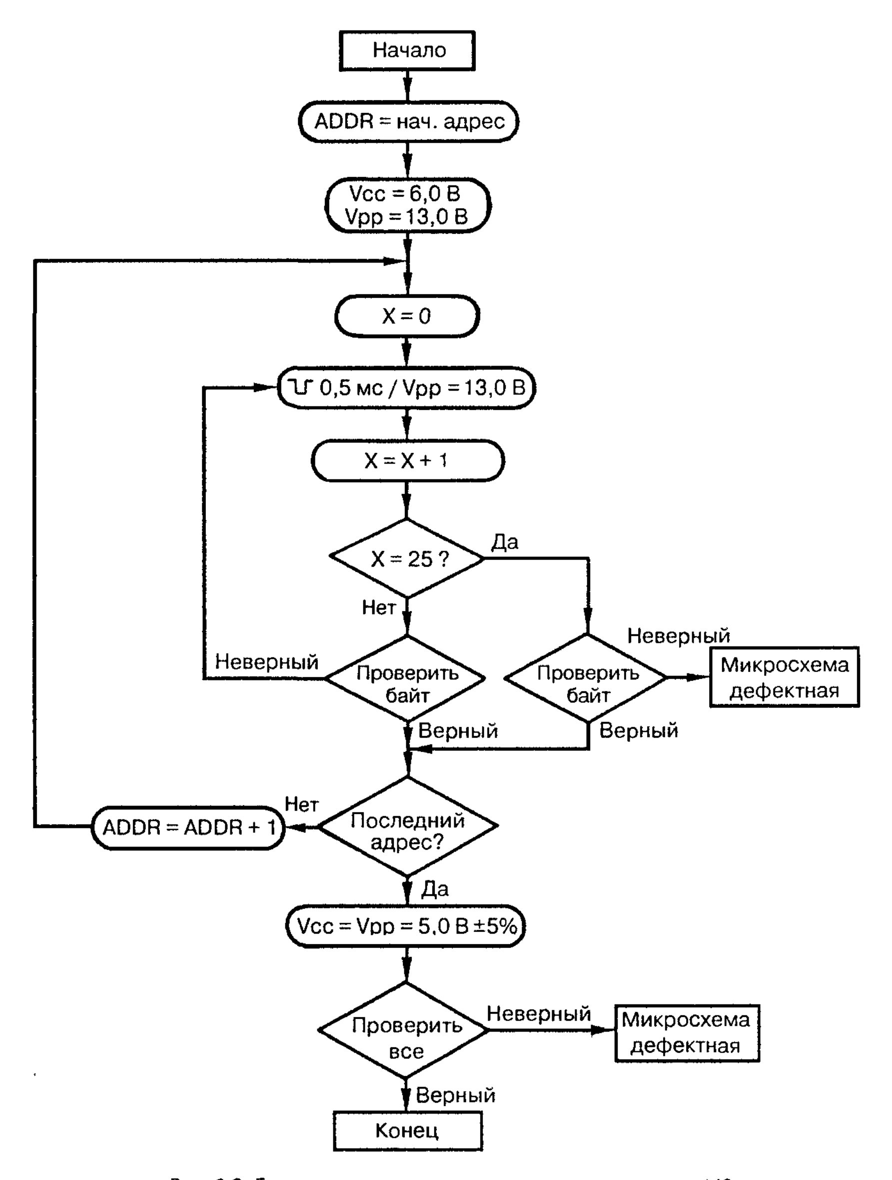


Рис. 2.9. Быстрый алгоритм, рекомендуемый компанией NS

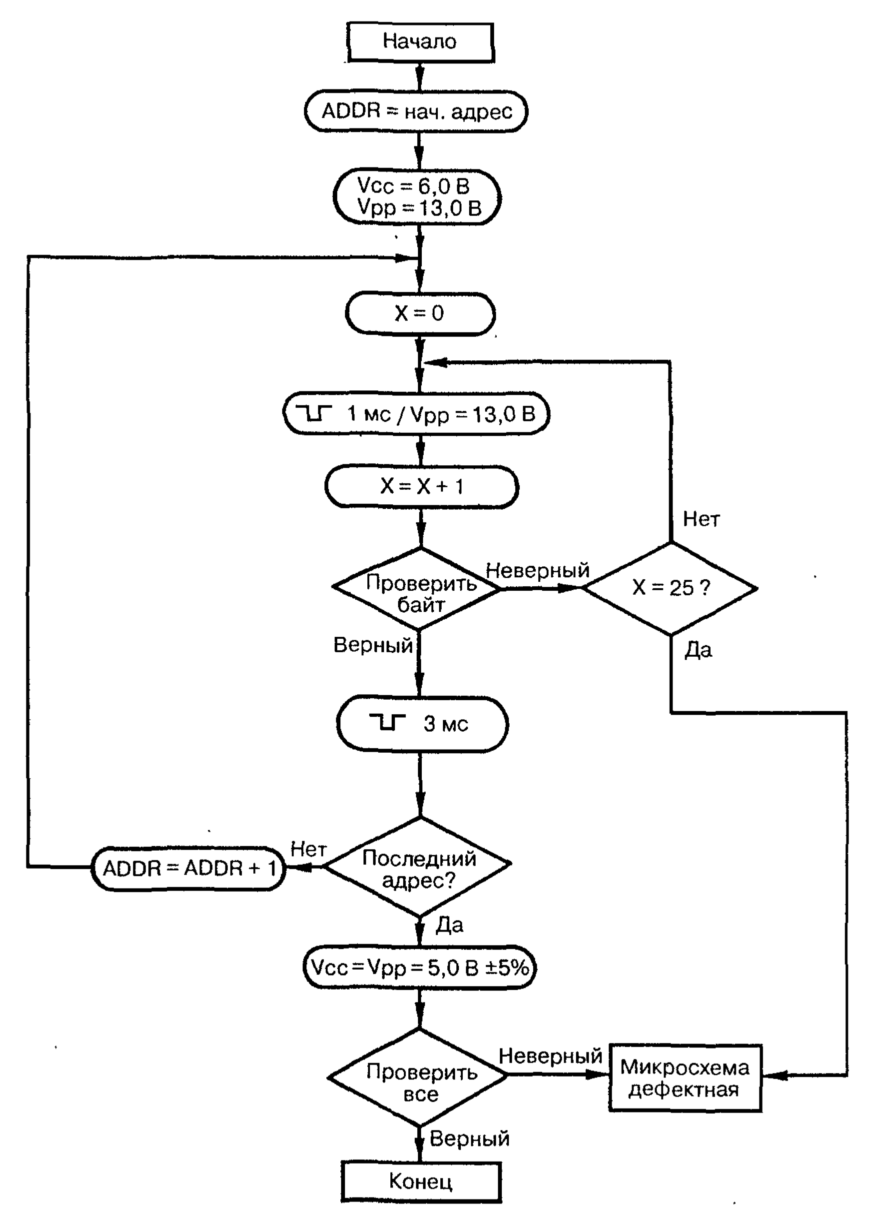


Рис. 2.10. Быстрый алгоритм, предложенный компанией Atmel

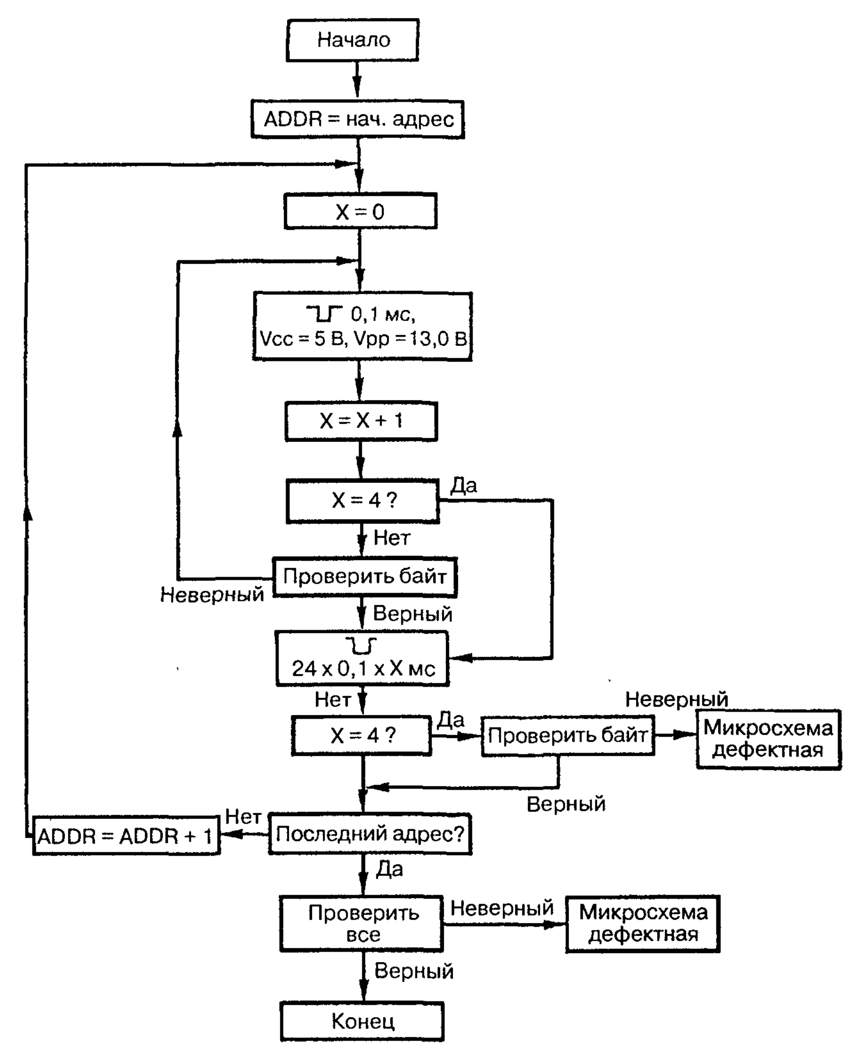


Рис. 2.11. Быстрый алгоритм, разработанный компанией Cypress

ПРОГРАММАТОРЫ

Запись данных в СППЗУ является довольно сложной процедурой, которая требует особого оборудования – программатора.

Такой прибор можно рассматривать как маленькое устройство для изготовления специальных интегральных микросхем, поскольку

программирование микросхемы СППЗУ превращает стандартную радиодеталь в организованный окончательным способом элемент, предназначенный для конкретного применения до тех пор, пока его содержимое не будет намеренно стерто или изменено пользователем.

Существует множество моделей и типов программаторов, предназначенных для применения в самых разных областях. Это разнообразие вызвано тем, что нужно либо многократно перепрограммировать один прототип в ходе разработки и отладки, либо изготавливать партии радиоэлементов в минимальные сроки.

Дубликаторы

Дубликаторы – особая категория программаторов, которые служат главным образом для производства большого количества идентичных радиоэлементов по одному исходному, служащему образцом.

Здесь очень важна производительность, поэтому такие аппараты выполняют, как правило, несколько копий одновременно (например, восемь). Кроме того, необходима систематическая и полная проверка запрограммированных радиоэлементов, чтобы исключить сбои и неполадки на следующих этапах производства из-за вероятных ошибок при их подготовке.

Промышленные программаторы обычно работают в автономном режиме: они самодостаточны, то есть для их запуска требуется только подключить питание.

Ручные программаторы

Ручной программатор – также автономный аппарат, обычно не имеющий никаких средств для связи с другим оборудованием или компьютерами. Работая как упрощенный дубликатор (одна копия за один цикл), он служит главным образом для быстрого программирования произвольных данных в ячейку по любому нужному адресу. Подобные программаторы снабжены набором переключателей для ввода данных и набора адреса, а также некоторым количеством индикаторов для контроля информации.

Ручные программаторы очень удобны на этапах разработки и отладки: прямо в процессе отладки можно наращивать содержимое СППЗУ бит за битом по мере написания или компилирования новых блоков данных или программ. Однако в данном случае разрешается преобразовывать только единицы в нули, а вот для выполнения обратной операции потребуется полностью стереть всю информацию в памяти.

Очевидно, что программировать подобным образом 8192 ячейки даже обычной микросхемы 27C64 слишком скучно, не говоря уже

о СППЗУ с большим объемом памяти. Но такой способ работы вполне приемлем для объемов данных в несколько сотен байтов, и этого количества вполне достаточно для ряда интереснейших разработок.

Программаторы для микро-ЭВМ

Микро-ЭВМ, особенно совместимые с ПК, стали очень распространенным инструментом, относительно дешевым и доступным. Их использование в качестве программатора СППЗУ благодаря специализированному программному и аппаратному обеспечению открывает широкие возможности для обработки информации и значительно облегчает и ускоряет процесс отладки даже самых сложных программ и систем.

Содержание СППЗУ любого объема может быть представлено в виде файла, записанного на дискету или жесткий диск и, следовательно, легко копируется и распространяется.

Такой файл разрешается создать, просмотреть или изменить с помощью многочисленных программ, уже существующих или специально для этого написанных.

Базы данных на различных электронных носителях дают возможность найти необходимую информацию о параметрах сигналов и алгоритмах программирования практически для всех разновидностей применяемых микросхем и соответственно выбрать для конкретной микросхемы наилучшие условия и режимы программирования.

Существует два больших семейства программаторов, приспособленных для работы с микро-ЭВМ: «внешние» и «внутренние».

Внешние программаторы

Внешний программатор – это более или менее автономный аппарат, снабженный всем необходимым для связи с микро-ЭВМ или IBM PC оборудованием. Как правило, связь организуется через последовательный порт RS232.

Файлы, содержащие данные для «зашивки» в то или иное СППЗУ, обычно с помощью компьютера «загружаются» в программатор и размещаются в специально предусмотренном для этого ОЗУ, данные из которого заносятся затем в одно или несколько СППЗУ.

И наоборот, содержимое СППЗУ, установленного в программатор, может быть считано и затем передано в компьютер, с помощью которого данные можно будет сохранить в файле на диске.

Между двумя аппаратами допустимо организовать и более развитый диалог: компьютер способен посылать программатору закодированные команды, а обратно получать различные сообщения.

Наиболее часто возникающее при этом неудобство связано с тем, что относительно медленная связь по последовательному каналу представляет «узкое место» в потоке информации, увеличивающее длительность процесса. Для передачи со скоростью 19200 бод 65536 бит данных для микросхемы 2764 потребуется едва ли не пять секунд – время, сравнимое с длительностью полного цикла программирования.

Приведенное выше замечание технически обосновано, но в большинстве случаев такие ограничения практически не сказываются на производительности. Кроме того, для подключения аппарата к компьютеру достаточно подсоединить кабель от программатора к свободному или же специально для этого освобожденному разъему (для программирования СППЗУ не нужны ни мышь, ни модем). Часто преимуществом будет и то, что для подключения не требуется отдельный слот ПК, которого может не быть, либо все они могут быть уже заняты. (Подобная ситуация возникает, кстати, довольно часто.)

Некоторые внешние программаторы способны использоваться автономно, если они снабжены клавиатурой и встроенным индикатором. Это очень удобно, например, при копировании, а также для выполнения модификации или прямого программирования некоторого разумного объема памяти.

Особенно представительный аппарат такого класса – программатор модели S4 английской компании Dataman: размером с большой калькулятор, он работает от встроенных аккумуляторов. Его можно повсюду брать с собой и подключать к ПК тогда, когда это действительно потребуется. Даже в автономном режиме это подлинная система разработки, мощная и гибкая. Программатор даже снабжен специальным кабелем для эмуляции микросхем, с помощью которого в процессе разработки и отладки он включается в схему вместо микросхемы СППЗУ. Другой подход состоит в том, чтобы применять разъем параллельного порта, к которому обычно подключаются принтеры стандарта Centronics. Этот разъем подойдет и для подключения некоторых программаторов, таким образом, последовательный порт RS232 остается свободным для других целей. При использовании параллельного порта скорость обмена информацией значительно выше.

Внутренние программаторы

Большое количество программаторов требуют установки в свободный слот ПК специальной платы, к которой через соединительный

кабель подключается выносной блок, снабженный панелькой, куда устанавливаются СППЗУ.

Существуют и дешевые модели, у которых панельки для микросхем СППЗУ установлены непосредственно на плате. Отметим, что работа с подобной конструкцией чрезвычайно неудобна.

Сообщаясь непосредственно с «сердцем» компьютера, эти устройства, очевидно, наиболее быстродействующие, но, к сожалению, без компьютера-«хозяина» они практически неработоспособны.

За счет развитого диалога, который можно установить с компьютером при подобной конструкции, у таких устройств гораздо больше различных функций, причем более сложных, чем у внешних программаторов.

Как правило, возможности хорошо продуманного внутреннего программатора зависят только от программного обеспечения: его настройка для работы с новыми микросхемами или алгоритмами потребует не больше времени, чем установка в дисковод дискеты для внесения дополнений, соответствующих новым данным, или регулярная загрузка этих дополнений по модему. Все это реально и не составляет большого труда по крайней мере до тех пор, пока существует изготовитель программатора.

В заключение хотелось бы порекомендовать доверять лишь тем изготовителям и поставщикам, которые имеют хорошую репутацию, и особенно не рассчитывать на тех, кто дает громкую рекламу неограниченной гарантии или обещает регулярные бесплатные обновления программного обеспечения.

Самодельные программаторы и наборы деталей для их изготовления

Программаторы промышленного изготовления отличаются своей надежностью в работе и обладают обширными возможностями.

Большинство изготовителей таких аппаратов поддерживают привилегированные отношения с поставщиками радиоэлементов и имеют, таким образом, детальную информацию по алгоритмам программирования, зачастую недоступную для конечного пользователя. Некоторые из таких изготовителей даже имеют статус «официально утвержденных» основными производителями программируемых радиоэлементов. Однако главный недостаток этих программаторов – дороговизна.

Более экономичен программатор собственной сборки, для чего можно либо купить набор деталей, либо изготовить все своими силами.

Существуют относительно недорогие наборы деталей, которые позволяют программировать современные СППЗУ самыми различными способами: либо полностью вручную, либо копированием образца, либо с помощью микро-ЭВМ или компьютера. Правда, обычно в таких программаторах нельзя использовать специальные скоростные или адаптивные алгоритмы, а программировать с их помощью можно только наиболее распространенные типы СППЗУ.

Далее в этой книге мы представим схемы очень простых программаторов, с помощью которых можно программировать самые различные семейства радиоэлементов, но только те, для которых алгоритмы программирования общедоступны. К счастью, таких микросхем достаточно, чтобы решить большинство практических задач, причем это именно те радиодетали, которые легче всего найти в розничной торговле и по вполне сходным ценам.

СТИРАНИЕ СППЗУ

Существует множество различных факторов, которые могут привести к полному или частичному стиранию данных, хранящихся в СППЗУ. На практике информацию, содержащуюся в этих радиоэлементах, стирают путем облучения их кристаллов ультрафиолетовыми лучами. При этом все биты во всех ячейках матрицы памяти переходят в состояние лог. 1.

Поэтому необходимо, чтобы корпус запоминающего устройства был снабжен специальным окошком, проницаемым для излучения с длиной волны 3000–4000 ангстрем; на которой процесс стирания идет наиболее эффективно.

Обычно микросхемы СППЗУ поставляются в керамических корпусах, оснащенных окошком из кварцевого стекла, но, к сожалению, такие корпуса очень дороги в производстве, и естественно, что микросхемы в таких корпусах также будут недешевы.

СППЗУ, предназначенные для серийного производства, часто не нуждаются в стирании, поэтому они выпускаются в пластмассовых корпусах, которые непроницаемы для ультрафиолетового излучения, но гораздо более дешевы.

Микросхемы СППЗУ в таких непрозрачных корпусах называются ОП и программируются только один раз. Однако всегда остается возможность преобразовать в лог. 0 биты, оставшиеся при первом программировании в состоянии лог. 1, поэтому в некоторых случаях их можно программировать несколько раз.

Для уверенного стирания информации в СППЗУ нужна определенная доза ультрафиолетового излучения, но следует помнить, что существует суммарная доза облучения, после которой запоминающее устройство должно рассматриваться как «полностью использованное». Поэтому очень важно строго следить за длительностью процесса.

Длительность стирания зависит от множества факторов: природных условий, степени старения микросхемы и свойств источника ультрафиолетового излучения, оптических свойств кварцевого стекла окошка СППЗУ, расстояния от СППЗУ до источника ультрафиолетового излучения и т.д.

В принципе для стирания вполне может хватить и солнечного света, поэтому для защиты от случайного стирания данных запрограммированного окончательным образом СППЗУ его окошко закрывают специальной светонепроницаемой наклейкой.

Наиболее часто в качестве источника ультрафиолетового излучения для стирания данных во всевозможных программируемых радиоэлементах используется так называемая бактерицидная флуоресцентная лампа, широко применяемая в различных приборах для стерилизации всевозможных материалов и дезинфекции помещений[1].

Подобные лампы похожи на классические лампы дневного света, для их эксплуатации необходимы такие же принадлежности (соединители, балласты, стартеры). Выпускаются бактерицидные лампы согласно стандартному ряду мощностей ламп дневного света. Как правило, в стирающих устройствах предпочитают использовать самые малогабаритные из доступных моделей, но среди радиолюбителей очень популярны модели TG 15 или TUV 15, мощностью 15 Вт и длиной 44 см.

Отметим также, что недавно появилась модель со встроенным балластом – TUV 6. Снабженная простым винтовым патроном, она питается непосредствено от сети 220 В, а потребляемая мощность составляет только 6 Вт.

Эта модель лучше всех остальных приспособлена для стирания СППЗУ и имеется на складе большинства компаний-дистрибуторов радиоэлементов; кроме того, эти компании предлагают и полные наборы деталей для сборки стирающих устройств, практически ничем не отличающихся от фабричных. Другой подход состоит в том, чтобы запастись всем необходимым у электрика и собрать схему, приведенную на рис. 2.12.

[1] В России подобные лампы использовались в кухонных вытяжках. – *Прим. ред.*

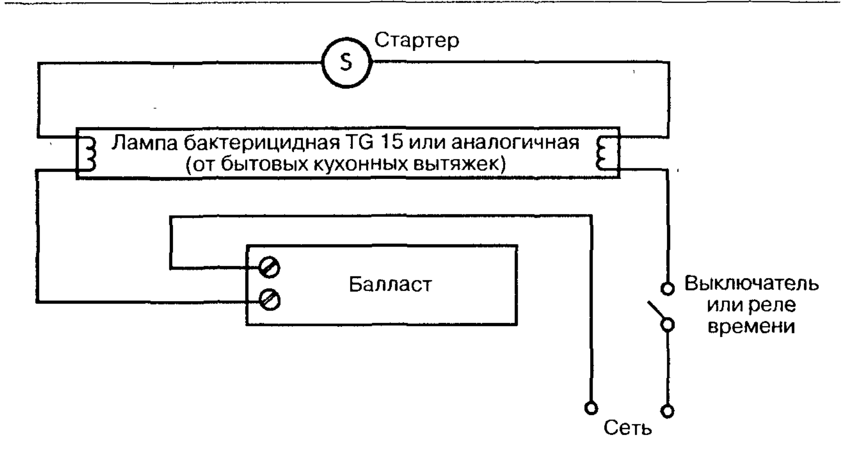


Рис. 2.12. Принципиальная электрическая схема стирающего устройства для СППЗУ

Процедура стирания требует облучения в течение 5–30 мин, поэтому полезно заранее установить реле времени, так как его использование устраняет любой риск превышения дозы облучения. Даже самая простая механическая модель (например, от старой стиральной машины) прекрасно справится с этой задачей.

Стираемые радиоэлементы необходимо разместить в корпусе так, чтобы их окошки были на расстоянии 2–3 см от источника ультрафиолетового излучения, причем желательно соединить между собой все выводы микросхемы, и самый простой способ для этого – наколоть микросхемы на антистатический проводящий пенополиуретан (см. рис. 2.13)[1].

Лампу желательно разместить над стираемыми компонентами, а не под ними, чтобы ограничить их нагревание. Нужно любым способом совместить достаточное вентилирование или обдув лампы с хорошей защитой от выхода ультрафиолетового излучения наружу, так как оно вредно для глаз и кожи. Наиболее часто используется конструкция, где микросхемы располагаются в выдвижном лотке, и лампу можно включить только тогда, когда лоток полностью задвинут в корпус.

[1] Этот материал используется для упаковки интегральных микросхем и других радиодеталей. Он выглядит, как поролон темно-серого цвета, бывает и мягким, и жестким; сопротивление на квадрат составляет от 10 до 1000 Ом. – *Прим. ред.*

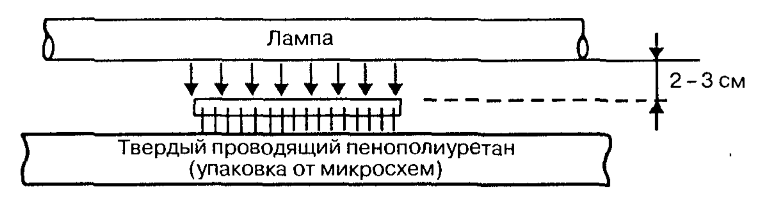


Рис. 2.13. Размещение СППЗУ относительно ультрафиолетовой лампы при стирании

Собранное стирающее устройство (или купленное) остается только откалибровать. Предварительно запрограммированное СППЗУ облучают поэтапно по пять минут с последующим «тестом на чистоту» – операцией, предусмотренной в большинстве программаторов. Впоследствии продолжительность облучения увеличивают в два раза по сравнению с той, которой во время калибровки хватило для полного стирания информации. Этого времени будет достаточно, чтобы окончательно стереть данные из запоминающего устройства. Обращаем ваше внимание на то, что периодически (допустим, два раза в год) следует проводить перекалибровку.

Отметим, что промышленностью выпускаются «пистолеты-стиратели», снабженные специальной лампой-вспышкой, излучающей ультрафиолетовые лучи. С помощью этих устройств можно стирать СППЗУ непосредственно в их панельках, причем всего за несколько десятков секунд.

ЭМУЛЯТОРЫ СППЗУ И ОЗУ С ЭЛЕМЕНТОМ ПИТАНИЯ

ОЗУ, выполненные по технологии КМОП, потребляют настолько мало энергии, что годами могут питаться от любой маленькой литиевой батарейки. Записанные в них данные становятся, таким образом, энергозависимыми, но возможность изменять в них информацию остается. Как и в обычных, стандартных статических ОЗУ, здесь можно не только изменить лог. 1 на лог. 0, но и наоборот, общее стирание информации становится ненужным.

Разрабатываются и серийно выпускаются микромодули, которые объединяют в одном специальном DIP корпусе КМОП ОЗУ, литиевую батарейку и схему управления питанием, обеспечивающую безопасность данных, например: ZEROPOWER, выпускаемые компанией SGS Thomson, NVSRAM компаний Benchmarq или Dallas, «эмуляторы СППЗУ», производимые фирмой Greenwich Instruments, и т.д.

В режиме считывания данных эти запоминающие устройства на 100% совместимы с СППЗУ эквивалентного объема. Допустимо, таким образом, подключить их к любой схеме и имитировать (эмулировать) СППЗУ. Естественно, не может быть и речи о том, чтобы программировать их так же, как СППЗУ, то есть высоким напряжением: нужно только подключить напряжение питания 5 В и подать на вывод $\overline{WR}$ отрицательный импульс после того, как будут выставлены адрес и данные.

Для завершения программирования достаточно нескольких сотен наносекунд, а это значит, что за доли секунды в такое запоминающее устройство можно записать несколько сотен килобитов. Это идеально при разработке и отладке приборов и систем, где требуются частые изменения данных. Нет ничего проще, чем перекопировать данные, окончательно записанные в ОЗУ с элементом питания, в СППЗУ с помощью практически любого программатора.

На рис. 2.14 приведена цоколевка микромодуля MK48Z02 компании SGS Thomson (2 Кб, совместима с микросхемами 2716), а на рис. 2.15 – цоколевка микромодуля MK48Z08 (8 Кб, совместима с микросхемами 2764).

Микромодуль GR281 компании Greenwich Instruments является аналогом микромодуля MK48Z02 компании SGS Thomson, а GR881 – аналогом MK48Z08.

За исключением вывода записи $\overline{WR}$, остальные выводы, в частности $\overline{OE}$ и $\overline{CE}$, используются точно так же, как и в стандартных микросхемах СППЗУ.

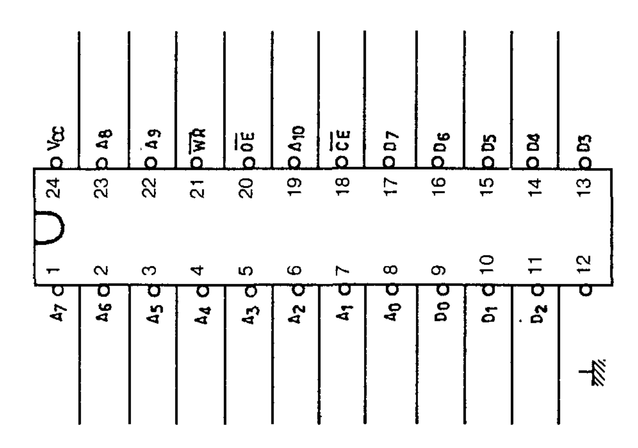


Рис 2.14 Цоколевка микросхемы 48Z02 (2 Кб)

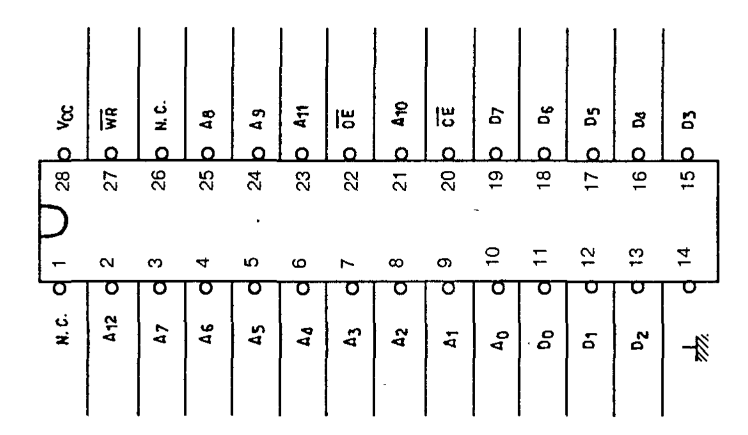


Рис. 2.15 Цоколевка 48Z08

ПОСЛЕДОВАТЕЛЬНЫЕ ЭСППЗУ

Технология ЭСППЗУ (или E2PROM) позволяет реализовать запоминающие устройства, которые находят применение в многочисленных областях, поскольку в них в определенной степени удалось соединить достоинства как СППЗУ, так и статических ОЗУ.

Рядом с целым семейством радиоэлементов с параллельным доступом, выполненных практически в полном соответствии со стандартом Bytewide, последовательные ЭСППЗУ (то есть с доступом по последовательному каналу) нашли широчайшее применение в самых разных областях – от периферийных устройств для ЭВМ до телевизоров, а также в чип-картах.

На данный момент последовательным ЭСППЗУ доступна память от 128 бит до 16 Кбит и более. Таким образом, они приближаются к старым параллельным СППЗУ небольших объемов.

Потоколы связи, как правило, классические и широко известные, в частности совместимые с I2C.

Концепция ЭСППЗУ

ЭСППЗУ (электрически стираемые перепрограммируемые ПЗУ) – это «постоянные» ЗУ, подобные СППЗУ, но информация в них стирается электрическими импульсами, а не ультрафиолетовыми лучами.

Однако ЭСППЗУ имеют огромное преимущество по сравнению с СППЗУ или даже с флэш-памятью, информация в которой может стираться не только полностью, но и частично. Стирание в современных моделях ЭСППЗУ осуществляется автоматически перед каждой

записью новых данных. Возникает вопрос: чем отличается ЭСППЗУ, например, от энергонезависимого ОЗУ со встроенным литиевым элементом питания?

В первую очередь длительностью операции записи. Как и в СППЗУ, данные в ЭСППЗУ записываются посредством воздействия высокого напряжения, то есть достаточно медленно. Это сходство может быть не вполне очевидно, так как программирующий импульс генерируется непосредственно в ЭСППЗУ встроенным источником высокого напряжения, выполненным по принципу емкостного преобразователя (накачка заряда), но в любом случае для записи каждого слова данных потребуется несколько миллисекунд.

Преимущество ЭСППЗУ заключается в том, что оно обеспечивает большую безопасность данных, чем ОЗУ с резервной батареей: достаточно запретить генерацию высокого напряжения, и любая запись, нужная или случайная, будет невозможна.

Обойти проблему большой длительности операции записи позволила разработанная недавно оригинальная технология. В микросхемах NOVRAM компании Xicor, например, каждая ячейка ЭСППЗУ дублируется ячейкой ОЗУ. Все происходит так, как будто в схеме работает обычное ОЗУ, но данные из него могут быть сохранены в ЭСППЗУ целым блоком, а в дальнейшем перезагружены обратно в ОЗУ. Процессы обмена данных могут инициироваться электрическими сигналами, программным обеспечением или автоматически – при отключении или подключении напряжения питания.

Другое существенное отличие между ОЗУ и ЭСППЗУ состоит в числе возможных циклов записи: в зависимости от марки микросхем, ошибки записи начинают появляться после определенного количества циклов стирания данных – от десяти тысяч до одного миллиона, в то время как в ОЗУ запись производится неограниченное число раз.

Эти цифры заметно превосходят соответствующие показатели для микросхем УФ СППЗУ, но они могут быть ещё улучшены методами коррекции ошибок или избыточностью, применяемыми либо в схеме, где используются обычные ЭСППЗУ, либо в самом ЭСППЗУ, куда встроены соответствующие блоки.

Например, использование кода Хэмминга позволяет увеличить срок службы ЭСППЗУ приблизительно в шесть раз, поскольку дефект практически всегда начинает сказываться только в одном из битов в целом слове. Таким образом, нелишним будет проведение контрольного считывания после каждой операции записи в любой схеме или системе, в которых требуется абсолютная целостность данных, записываемых в запоминающее устройство.

Значительные успехи были достигнуты и в повышении гарантированной длительности хранения данных в ЭСППЗУ, по этому параметру они часто даже лучше, чем микросхемы УФ СППЗУ и ОЗУ с элементом резервного питания: они работают от десяти лет и до века, в зависимости от марки.

Хотя за последние годы многое в электронике изменилось в лучшую сторону, не стоит абсолютно полагаться на элементы резервного питания и его контакты, особенно с точки зрения долговременной стабильности и надежности в тяжелых условиях эксплуатации. Проблемы такого рода у ЭСППЗУ отсутствуют. Кроме того, пока трудно себе представить, как можно установить элемент резервного питания в корпусе DIP с восемью выводами и тем более в корпусе для поверхностного монтажа (SMD).

ЭСППЗУ с последовательным доступом

Хорошо известно, что каждый дополнительный вывод интегральной микросхемы стоит дорого: высокая цена обусловлена и самим радиоэлементом, и размерами и сложностью печатной платы, на которую будет установлена эта микросхема.

Хотя ЭСППЗУ значительно дороже, чем СППЗУ с эквивалентным объемом памяти, последовательный интерфейс позволяет получить существенную экономию.

Последовательная связь между микросхемами только по двум или трем проводникам уменьшает скорость обмена, но оказывается, в конце концов, выгодной в большинстве областей применения. Всемирный успех шины I2C убедительно свидетельствует об этом.

Добавим, что некоторые приложения были изначально рассчитаны на такой интерфейс, например чип-карты с восемью или даже только с шестью контактами.

Кажущаяся сложность обмена нужными данными сначала препятствовала успеху ЭСППЗУ с последовательным доступом, и радиолюбители продолжали использовать параллельные запоминающие устройства стандарта Bytewide даже для программирования всего нескольких байтов. Но на сегодняшний день посылка или прием «последовательных» данных через порт микропроцессора воспринимается как простое и вполне обычное действие.

Подобные предпочтения вполне понятны: маленький корпус с восемью выводами может сохранить на сотню лет столько же данных, сколько и микросхема 2716, работает он только от +5 В (или даже +2,5 В), из него можно считывать и полностью или частично изменять записанные данные при помощи всего нескольких команд, передаваемых по

простой двух- или трехпроводной шине; кроме того, он располагает защитой от записи. Именно поэтому все большее число компаний производителей, учитывая растущий успех таких радиоэлементов, предлагают последовательные ЭСППЗУ.

Рабочие параметры последовательных ЭСППЗУ в зависимости от марки заметно различаются, в частности напряжением питания, сроком службы и длительностью хранения данных. Обычно эти характеристики имеют следующий вид: от 5 В/10000 записей/10 лет до 2,5 В/1000000 записей/100 лет, со всевозможными промежуточными сочетаниями. Таким образом, выбор зависит от предполагаемой области применения, реальной стоимости и требуемой надежности.

Микросхемы разных изготовителей, имеющие одно и то же обозначение (например, очень распространенное – 93С06), могут абсолютно отличаться друг от друга: цоколевкой, структурой и даже системой команд. Поэтому еще большую осторожность следует проявлять при выборе изделий от «вторых поставщиков», причем все эти нюансы желательно разрешить в начальной стадии разработки проекта.

Мы, естественно, не можем описывать здесь все разновидности микросхем и перечислять все компании, производящие последовательные ЭСППЗУ. Отметим только, что существуют два больших семейства последовательных ЭСППЗУ, которые, кажется, просто созданы для того, чтобы конкурировать друг с другом: ЭСППЗУ с двухпроводной шиной – стиль I2C, и ЭСППЗУ с трехпроводной шиной – стиль Microwire или SPI (шина, изначально предназначенная для микроконтроллеров 6805 и 68HC11 компании Motorola).

Термины I2C и Microwire в действительности являются зарегистрированными торговыми марками, первая принадлежит фирме Philips, а вторая – компании National Semiconductor. Только их владельцы или производители, заранее купившие лицензии, могут официально упоминать и использовать эти термины.

Наиболее ясно обстоит дело с шиной I2C. Многие компании приобрели лицензию и могут использовать такую аббревиатуру при условии соблюдения технических требований, связанных с этой шиной и ее протоколом. Подобные радиоэлементы должны нормально и без проблем функционировать в любой системе с шиной I2C.

Некоторые фирмы не желают приобретать лицензии I2C и поэтому не могут продавать свою продукцию под этой маркой. Несмотря на наличие аббревиатур SCL и SDA для обозначения линий их шины, совместимость с официальными требованиями I2C часто только

приблизительна, а иногда и просто отсутствует. Попытка использования таких радиоэлементов с настоящей шиной I2C может привести к серьезным конфликтам в системе.

Но тем не менее существуют превосходные изделия, способные надежно работать при условии применения соответствующего именно им протокола связи, очень близкого к I2C, но немного от него отличающегося и именно поэтому не попадающего под действие лицензии.

ЭСППЗУ с шиной типа I2C

В табл. 2.6 приведены наиболее распространенные типы микросхем ЭСППЗУ с шиной I2C или ей подобными интерфейсами.

Учитывая требования стандарта I2C, все эти запоминающие устройства имеют организацию по восемь бит на слово, причем объемы памяти большие 256 байт часто разбиваются на столько «страниц» по 256 байт, сколько необходимо.

Таким образом, наиболее распространенный объем памяти составляет 256 байт, хотя теперь он превышает, благодаря различным программным способам управления, даже теоретическую границу в 2 Кб (8 блоков по 256) или 16 Кбит. На рис. 2.16 показана цоколевка микросхем памяти ЭСППЗУ I2C.

Назначения четырех выводов в принципе одинаковы почти для всех типов и разновидностей практически у всех производителей:

(4) V_{SS} (общий);
(5) SDA (данные);
(6) SCL (синхронизация);
(8) V_{CC} (питание).

A_0 1 — 8 V_{CC}
A_1 2 — 7 Зависит от типа
A_2 3 — 6 SCL
V_{SS} 4 — 5 SDA

Рис. 2.16. Цоколевка ЭСППЗУ I2C

Выводы 1, 2 и 3 выделены для трех младших разрядов I2C-адреса радиоэлемента (A0, A1, A2), четыре старших разряда всегда 1010, самый младший бит адреса служит указателем режима чтения-записи ($R\overline{W}$).

Следовательно, можно организовать нормальную работу на одной и той же шине до восьми идентичных радиоэлементов, так как каждому из них разрешается присвоить уникальный адрес.

Однако это не относится к семейству микросхем 24C00, 24001 и 24C01 компании Xicor, для которых нет возможности установить I2C-адрес, соответственно используются только непосредственно адреса байта памяти, выводы 1–3 не задействованы.

Такая организация (не согласующаяся с требованием I2C) позволяет упростить диалог, но исключает совместную работу нескольких радиоэлементов на одной шине. Однако в простых устройствах это может и не быть серьезным недостатком.

Таблица 2.6. Панорама основных типов ЭСППЗУ с шиной I2C

I2C и их аналоги (двухпроводная шина)				
Организация	**SGS-THOMSON**	**XICOR**	**ATMEL**	**MICROCHIP**
16×8		24C00 24001		
128×8	24C01	24C01 24C01A 24012	24C01	85C72 24C01A 24LC01 24LH01
256×8	25C02A 24C02A	24C02 24022	24C02	85C82 24C02A 24LC02 24LH02
2×256×8	25C04 24C04	24C04	24C04	85C92 24C04A 24LC04 24LH04
4×256×8	24C08	24C08	24C08	
8×256×8		24C16 24164	24C16	24C16 24LC16 24LH16

Микросхемы 24C01A (компании Xicor или Microchip), напротив, используют все три вывода A0, A1, A2 и учитывают их состояние при формировании своего адреса «раба», в то время как микросхема 24C01 компании Atmel эти выводы имеет, но в адресе их не учитывает.

Назначение вывода 7 меняется в зависимости от производителя микросхемы: он не используется у моделей 24C00, 24001, 24C01, 24012 компании Xicor и у прибора 85C72 компании Microchip, но должен быть заземлен у микросхем 24C01 производства Atmel, 24C01A или 24LC01 производства Microchip.

У микросхемы 24C01A компании Xicor, напротив, заземление вывода 7 разрешает запись, в то время как подача на него высокого логического уровня запрещает ее, блокируя встроенный генератор напряжения программирования.

У микросхем 24C02A и 25C02A производства SGS-Thomson с помощью вывода 7 допустимо выбрать число байтов, которые будут задействованы в одной операции записи (4 или 8).

У приборов 24C02 компании Xicor и 24LC02 производства Microchip с помощью вывода 7 можно разрешить или запретить запись во всем запоминающем устройстве, тогда как у микросхемы 24C02A производства Microchip эта защита распространяется только на верхнюю половину памяти.

Начиная с объема 512×8 бит, матрица памяти разделена на блоки по 256 бит, которые нужно суметь выбрать в адресе «раба». Для этого пользуются одним, двумя или тремя битами, где обычно фиксировалось состояние на выводах A0, A1 и A2.

У запоминающих устройств с объмом памяти 4 Кб (2×256×8 бит) только состояние выводов A1 и A2 влияет на адрес радиоэлемента. Поэтому не более четырех подобных запоминающих устройств могут сосуществовать на одной шине, а освобожденный таким образом бит P0 позволяет выбрать ту или иную из двух страниц по 256 байт.

Если вывод 1 (прежде A0) свободен, то в большинстве случаев его нужно соединить с общим проводом, но можно оставить свободным, как у микросхем 24C04 компании Atmel.

У приборов 24C04 и 25C04 (работающих от напряжения 2,5 В) производства SGS-Thomson вывод 1 служит для того, чтобы задействовать достаточно сложную систему защиты от записи, которую мы опишем только кратко.

Когда на выводе 1 высокий логический уровень, в последнем байте запоминающего устройства (адрес FFh на странице P0=1) размещается специальный защитный регистр, который в действительности определяет границу между зоной, защищенной от записи, и зоной, доступной как для записи, так и для чтения.

Если вывод 1 подключен к общему проводу, то вся память обычно доступна для записи и чтения, что позволяет, в частности, изменить содержимое спецрегистра, который сам размещен в защищенной зоне.

У запоминающих устройств с объемом памяти 8 Кбит (4 страницы по 256 байт) только состояние вывода A2 используется при установлении адреса радиоэлемента, таким образом освобождаются разряды P0 и P1 для программного выбора страниц.

И наконец, у запоминающих устройств на 16 Кбит адрес нельзя определять извне: чтобы различать 8 страниц по 256 байт, нужно три разряда (P0, P1, P2).

Действительно, подключение одного 16-килобитного кристалла к шине I2C практически эквивалентно подключению восьми кристаллов по 2 килобита: адресация происходит тем же способом, но с выигрышем в компактности.

Микросхема 24164 компании Xicor занимает отдельное место в этом семействе. Вместо того чтобы оставаться неиспользованными, выводы 1, 2 и 3 служат здесь для изменения определенной стандартом для различных по назначению радиоэлементов части адреса. Только один старший разряд этого адреса остается логической единицей,

а остальные три зависят от уровней, приложенных к выводам S0, $\overline{S1}$ и S2. Если эти три вывода заземлены, адрес получается стандартным – 1010, обычно он применяется в I2C для запоминающих устройств.

Следует отметить, что адрес 1010 был определен на международном уровне Комитетом I2C для запоминающих устройств всех типов. Использование для них другого адреса неизбежно приводит к риску конфликта с другими радиоэлементами I2C, подключаемыми к той же шине.

Хотя эта особенность интегральной микросхемы 24164 может оказаться полезной в некоторых случаях, ясно, что данный радиоэлемент совершенно «вне закона» с точки зрения стандарта I2C. Но действительно, почему бы таким и не быть, если производителю не требуется аббревиатура I2C?

Не вдаваясь в подробности, напомним, что обмен данными вызывается определенным числом логических условий на линиях SCL и SDA:

- шина не занята (на линиях SCL и SDA высокий уровень);
- Start-условие или начало обмена (спадающий фронт на SDA, когда на SCL высокий уровень);
- Stop-условие или конец обмена (нарастающий фронт на SDA, когда на SCL высокий уровень);
- данные готовы (на SDA неизменный уровень, в то время как на SCL высокий уровень);
- успешный прием или ACK (SDA переводится на низкий уровень, когда по линии SCL передается девятый тактовый импульс);
- запись (младший разряд адреса в состоянии лог. 0);
- чтение (младший разряд адреса в состоянии лог. 1).

ЭСППЗУ MICROWIRE ИЛИ ИХ АНАЛОГИ

В табл. 2.7 представлены наиболее распространенные ЭСППЗУ с шиной Microwire и совместимые с ней.

Первоначально оптимизированные для работы в устройствах с микроконтроллерами COP компании NS (National Semiconductor), эти запоминающие устройства обычно организованы в регистры по 16 разрядов.

Однако некоторые модели могут быть переорганизованы в систему с удвоенным числом регистров по 8 разрядов.

Объемы памяти здесь более скромные, чем у запоминающих устройств для шины I2C, одна из наиболее популярных конфигураций – 16×16 бит (или всего лишь 32 байта), которая особенно дешева.

Отметим, что этих 256 бит (объем памяти кристалла для телефонных чип-карт) оказывается вполне достаточно для решения многих

Таблица 2.7. Основные типы ЭСППЗУ Microwire

Microwire и их аналоги (трехпроводная шина)					
Организация	**NS**	**SGS-THOMSON**	**XICOR**	**ATMEL**	**MICROCHIP**
16x16	9306/7 93C06 93CS06	93C06	24C44 24C45 (NOVRAM)	93C06	
32x8		93C06			
32x16	93CS26				
64x16	9346 93CS46	93C46 93CS46 93CS47		93C46	93C46 59C11 93LC46
128x8		93C46		93C46	59C11 93LC46
128x16	93CS56	93CS56 93CS57		93C56	93C56 93LC56
256x8				93C56	93C56 93LC56
256x16	93CS66			93C66	93C66 93LC66
512x8				93C66	93C66 93LC66

задач, в частности для идентификации: простая память 9306, например, может содержать 16 кодов, каждый из которых способен принимать 65536 возможных комбинаций.

К данной категории также принадлежат микросхемы NOVRAM 24C44 и 24C45 autostore (автосохранение) компании Xicor. Для этих изделий разработано множество вариантов применения.

Шина Microwire обычно представляется как трехпроводная, если не считать общий провод и питание: выбор кристалла ($\overline{CS}$), синхронизация (SK), данные.

Для передачи данных предусматриваются две линии: ввода (DI) и вывода (DO), что напоминает четырехпроводную шину, но на самом деле линии DI и DO часто объединяют на одном и том же проводе, хотя при этом и необходима особая осторожность во избежание конфликтов на шине.

Цоколевка этих радиоэлементов практически стандартизирована, по крайней мере в том, что касается шести из восьми выводов корпуса (см. рис. 2.17).

Именно из-за различного назначения выводов 6 и 7 микросхемы разной организации или разных производителей могут не совпадать друг

CS 1 — 8 V_{SS}
SK 2 — 7 } Зависит от типа
DI 3 — 6
DO 4 — 5 V_{SS}

Рис. 2.17. Цоколевка ЭСППЗУ Microwire

с другом по цоколевке. В наиболее простых моделях (NMC 9306, NMC 9346 и т.д.) эти два вывода просто не используются. У приборов NOVRAM компании Xicor они служат для того, чтобы непосредственно управлять процессами сохранения и обмена данными между блоками ОЗУ и ЭСППЗУ.

Встречаются и следующие назначения этих выводов, каждое из которых соответствует определенной модификации или исполнению:

- BPE (Bulk Programming Enable) позволяет разрешить или запретить запись и стирание всего содержимого памяти целиком;
- PRE (Protect Register Enable) позволяет разрешить или запретить доступ к регистру защиты, содержание которого определяет адрес первого регистра зоны, защищенной от записи;
- PE (Program Enable) позволяет разрешить или запретить запись и стирание;
- ORG (ORGanisation) служит для определения организации запоминающего устройства в регистры по 16 или 8 бит;
- TEST применяется для тестирования на заводе; использовать его следует согласно требованиям изготовителя (как правило, его надо оставлять свободным или заземлять).

В зависимости от типономиналов и марок возможны различные варианты назначения этих выводов и даже различные наборы команд. Действительно, обмен данными с запоминающими устройствами по шине Microwire более сложный (но также и более богатый), чем с устройствами на шине I2C: запоминающее устройство ожидает на своем выводе DO сообщение, состоящее из стартового бита (1), кода операции (опкода), определяющего выполняемое действие, и адреса регистра (если таковой необходим), а затем – данных.

Что касается команды чтения, то считываемые биты появятся на выводе DO сразу после завершения посылаемого на вывод DI сообщения (если оно было опознано). Все процессы выполняются по сигналу синхронизации, поступающему по линии SK.

Очевидно, что формат этих обменов более или менее зависит от типа микросхемы и от ее изготовителя, в частности от числа разрядов, составляющих адреса. Таким образом, настоятельно рекомендуется получить и подробно изучить детальное описание каждого радиоэлемента, с которым предполагается работать.

Для микросхем с одним и тем же названием коды команд также могут немного различаться у разных изготовителей (например, у компаний NS и Microchip).

ГЛАВА СТРАНИЦА

3 ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ СХЕМЫ

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ СХЕМЫ

Программируемые логические интегральные схемы (ПЛИС – Programmable Logic Device, PLD) основаны на технологиях, подобных тем, что используются в программируемых запоминающих устройствах, но применяются эти технологии в несколько необычных целях.

ПЛИС представляет собой кристалл, содержащий большое число элементов, реализующих базовые логические функции (главным образом вентили и триггеры). Эти функции пользователь может организовать в ту или иную логическую схему в соответствии со своими требованиями, причем он может произвольно «разводить» не только соединения между логическими и/или буферными элементами, но и определять назначение выводов кристалла и корпуса микросхемы. Понятно, что такая технология дает возможность реализовывать логические схемы, которые полностью соответствуют конкретному проекту или разработке.

Если такая организация производится с помощью маски во время изготовления схемы (gate array – базовый матричный кристалл, БМК), это выгодно только для крупносерийного производства. Кроме того, предполагается большой промежуток времени между окончанием разработки схемы и моментом предоставления первых опытных образцов, а любая ошибка или изменение в схеме стоят очень дорого.

Но пользователь может сам организовывать свои логические схемы, причем даже в единственном экземпляре, при помощи простого и дешевого по сравнению с программаторами ЭСППЗУ оборудования. И это оказывается гораздо удобнее. Запрограммированную таким образом в микросхему «таблицу соединений» частного применения можно легко стереть ультрафиолетовым излучением или электрически, а затем перепрограммировать на новую. Немаловажно и то, что можно предусмотреть защиту от копирования схемы специальным «битом защиты».

Все это вполне реально и достижимо. В настоящее время выпущено множество вариантов мощного программного обеспечения, ориентированного на совместимые ПК, причем эти системы иногда бесплатны, и с их помощью выполняется собственно программирование схемы согласно описанию логических функций и уравнений, разрабатываемых пользователем при помощи компьютерного «логического моделирования», при котором проводится также автоматическая проверка правильности подготовленных данных.

Сегодня уже реально написать нечто похожее на «программу», а затем на ее основе без особого труда получить специальную интегральную микросхему. Хотя некоторые изготовители всячески стараются опровергнуть правильность этой идеи, но самостоятельно сконструировать подходящий программатор вполне возможно, а в некоторых случаях и удивительно просто.

Для радиолюбителей-конструкторов такой подход дает целый ряд преимуществ: выигрыш в размерах и простоте печатных плат, ускорение процесса разработки новых конструкций, повышение надежности, уменьшение количества нужных типов стандартных цифровых микросхем, повышение собственной квалификации, защита от копирования схем.

Покупатель готовых устройств также может воспользоваться подобными преимуществами, но он должен осознавать, что в вопросах обслуживания, ремонта или модификаций он «связан по рукам и ногам»: у него не будет доступа к наиболее важным частям схемы, а купить запасные части он сможет только те, которые ему пожелает предоставить производитель.

БАЗОВЫЕ СТРУКТУРЫ

Каждый производитель программируемых интегральных микросхем, вступая в конкурентную борьбу, старается использовать свое собственное «секретное оружие» и предлагает вниманию пользователя исключительные стороны своей продукции. Но в основе часто лежат одни и те же базовые идеи и принципы – настолько часто, что можно говорить о совместимости большинства выпускаемых версий.

С определенной долей условности запоминающие устройства ППЗУ или СППЗУ допустимо рассматривать как программируемые логические схемы: в режиме чтения запоминающее устройство имеет некоторое количество входов (шина адреса) и некоторое количество выходов (шина данных).

Для каждого сочетания входных состояний можно определить некоторую комбинацию состояний выхода: достаточно записать необходимые данные по соответствующим адресам.

Например, можно запрограммировать СППЗУ таким образом, что оно будет выполнять функции преобразователя двоичного кода в 7-сегментный, в частности когда таблица истинности стандартной модели не удовлетворяет требованиям проекта. Понятно, что в этом случае используют только четыре линии адреса и семь линий данных

и программируют только 16 ячеек по соответствующим адресам – ничтожно малую часть доступного объема памяти.

С точки зрения алгебры логики (булевой алгебры), СППЗУ можно рассматривать как набор логических элементов И (на каждый адрес – по одному элементу со столькими входами, сколько имеется адресных линий), после которых следует программируемая матрица логических элементов ИЛИ (на каждую линию данных – по одному элементу, снабженному столькими входами, сколько имеется адресов).

Простая память 27C64, например, эквивалентна 8192 вентилям И с 13 входами, за которыми следует 8 вентилей ИЛИ с 8192 входами. Причем между этими двумя схемами есть 65536 перемычек, которые можно индивидуально «снимать» или «ставить», программируя микросхему.

Чтобы нарисовать соответствующую схему, необходимо оговорить некоторые упрощенные обозначения. На рис. 3.1 представлен пример схемы некоего СППЗУ, оснащенного только четырьмя адресными линиями и четырьмя линиями данных.

Это устройство соответствует схеме с 4 элементами ИЛИ (каждый с 16 входами) и с 16 вентилями И (с 4 входами каждый), связанными между собой коммутационной матрицей с 64 узлами. Все входы одного вентиля представлены одной линией, вертикальной для элементов ИЛИ и горизонтальной для элементов И. Они пересекают под прямым углом линии, с которыми могут быть соединены: выходы элементов И со входами элементов ИЛИ и выходы входных буферов со входами элементов И.

Разумеется, входы элементов И могут быть подключены как к прямому, так и к инверсному выходу каждого из входных буферных элементов.

Соединения между вертикальной и горизонтальной линиями обозначены крестом или точкой. Эти соединения, очевидно, неизменны на уровне входов элементов И (они определяют архитектуру запоминающего устройства), но коммутационное поле между элементами И и элементами ИЛИ полностью программируется пользователем. В нашем примере задействованы все возможные связи, что соответствует случаю чистого СППЗУ (все биты имеют значение 1).

Программирование заключается в том, чтобы избирательно уничтожить эти своего рода «плавкие перемычки», всем разрушенным перемычкам соответствует 0, записанный в определенную ячейку памяти. СППЗУ, таким образом, служат для того, чтобы синтезировать любые разновидности чисто комбинаторных схем, но часто за счет

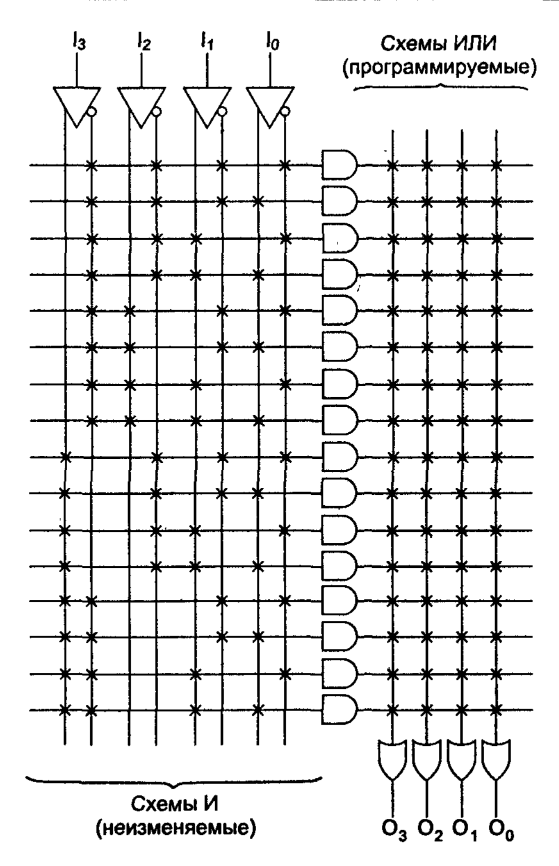


Рис. 3.1. Архитектура (логическая структура) СППЗУ

неполного использования всех возможностей радиоэлемента. Кроме того, подготовка данных для программирования не всегда будет легким занятием (нужно «обсчитать» 8192 байт для полного контроля за 13 входными линиями и 8 выходными).

Рис. 3.2 иллюстрирует другой подход к этой проблеме: архитектура ППЛМ (программируемая пользователем логическая матрица – Field Programmable Logic Array, FPLA), в которой коммутационное поле между входными буферами и элементами И также программируется.

Такое решение дает большую гибкость, позволяя лучше использовать внутренние ресурсы. В заключение отметим, что затраты труда и времени на разработку остаются достаточно большими.

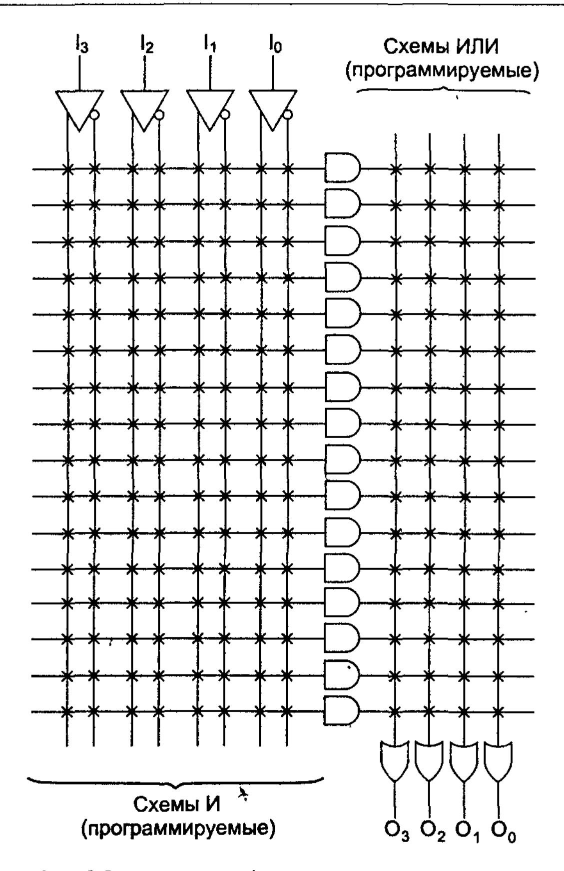


Рис. 3.2. Архитектура (логическая структура) ППЛМ

Основываясь на подобных базовых структурах, изготовители интегральных микросхем предлагают бесчисленные варианты архитектуры и топологии, более или менее совместимые между собой, которые расширяют области применения программируемой логики. Существенное улучшение состоит в том, что перед выводами кристалла размещают не простой двунаправленный буферный элемент, а универсальную многофункциональную самопрограммируемую макроячейку.

Некоторые, на сегодняшний день наиболее совершенные в техническом плане программируемые логические схемы только отдаленно напоминают своих далеких предшественников. Теперь производители соперничают друг с другом в основном по части воображения: кто

включит больше всевозможных программируемых логических функций в коммутационные матрицы, которые сами по себе также программируются пользователем. Высококлассные радиоэлементы, предлагаемые такими компаниями, как Altera, Atmel, Intel, ICT, Xilinx, Actel, Lattice, AMD, все более широко распространяются и находят применение в самых разных областях.

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ МАТРИЦЫ

Понятие «ПЛМ» (программируемые логические матрицы Programmable Array Logic) совсем не ново: оно было введено в 1970-х годах компанией Monolithic Memories (MMI). С тех пор этот термин используется многочисленными производителями, иногда изменяющими, и очень существенно, свои продукты. Внутренняя архитектура схем ПЛМ вытекает из известного положения, согласно которому любая функция комбинаторной логики может быть сведена к сумме произведений (напоминаем, что термин «сумма» условно обозначает операцию «логическое ИЛИ», а термин «произведение» – операцию «логическое И»).

Классические средства, такие как теорема де Моргана, метод Квайна Мак-Класки или таблицы (карты) Карно, дают возможность получить минимизированные булевы уравнения из полных уравнений или таблиц истинности, выведенных непосредственно при изучении реализуемой логической схемы.

Схема ПЛМ состоит из следующих логических блоков:

- некоторое количество входных выводов, каждый из которых оснащен буферным элементом, имеющим одновременно и прямой, и инверсный выходы;
- программируемая матрица логических элементов И с многочисленными входами, причем на каждый вход каждого логического элемента И может быть подан через повторитель или инвертор любой входной сигнал или его дополнение через программируемое коммутационное поле;
- жесткая (неизменная) матрица логических элементов ИЛИ с определенным количеством входов, каждый из которых подключен к соответствующему числу выходов вентилей И;
- в случае необходимости несколько буферных схем, размещенных между выходами вентилей ИЛИ и внешними выводами ПЛМ: здесь можно использовать инверторы, буферы с третьим состоянием или D-триггеры. В последнем случае ПЛМ служит

для того, чтобы реализовывать не только комбинаторные схемы, но и относительно сложные последовательные системы, например счетчики или регистры;
- обычно добавлено несколько ячеек обратной связи, возвращающих выходные сигналы в программируемое коммутационное поле для того, чтобы они могли быть обработаны как входные сигналы.

Рис. 3.3 показывает, что такая структура в действительности полностью противоположна строению СППЗУ, рассмотренному выше.

Каждый вентиль И имеет столько же входов, сколько вертикальных линий насчитывается в матрице (обычно 32, иногда больше), но каждый вход может быть подключен к вертикали, которая для него выделена только посредством плавкой перемычки.

Первоначально целая, она может быть разрушена извне во время операции программирования. Такими необратимо разрушаемыми плавкими перемычками действительно оснащались первые, биполярные ПЛМ, но теперь они заменены МОП транзисторами с плавающим затвором: вместо того чтобы физически сжигать перемычку, просто инжектируют заряд на плавающий затвор транзистора, как и в СППЗУ. Хотя этот заряд способен существовать годами, его можно быстро удалить либо электрически, либо облучением ультрафиолетом, если ПЛМ размещена в корпусе с кварцевым окошком. В таком случае мы располагаем стираемыми и перепрограммируемыми ПЛМ, принадлежащими к семействам УФ СПЛМ или ЭСППЛМ (стираемые ПЛМ – Erasable PLD, EPLD). Это идеальный вариант при разработке и отладке, которые, как правило, требуют нескольких попыток.

Размещенные в пластмассовых корпусах, непрозрачных, но исключительно дешевых, УФ СПЛМ, как и СППЗУ, лишены возможности стирания, при этом называться они будут ОП (однократно программируемые – One Time Programmable, OTP).

Выходы логических элементов И называются термами произведений и объединены в группы (обычно по восемь) на уровне входов логических элементов ИЛИ, выходы которых подключены к выводам ПЛМ. Именно в элементах ИЛИ осуществляется суммирование окончательных результатов.

Простейшая ПЛМ часто содержит 8 логических элементов ИЛИ с 8 входами, то есть 64 терма произведений из 64 логических элементов И с 32 входами каждый, а также 8 входных парафазных буферов

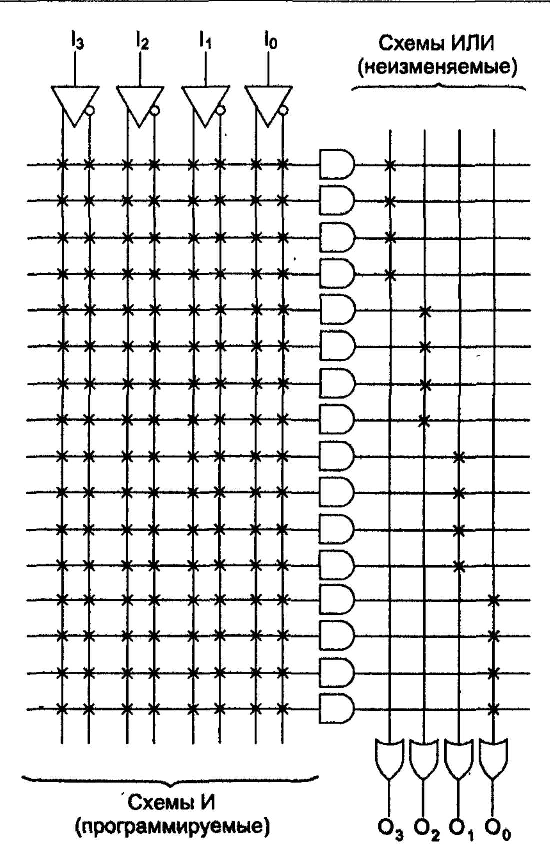


Рис. 3.3. Архитектура (логическая структура) ПЛМ

и 8 выходных схем (инверторы, буферы, триггеры и т.п.). Это соответствует 2048 плавким перемычкам.

Внутренние схемы подобных радиоэлементов будут достаточно сложны, если не использовать оговоренные выше условные обозначения, которые мы уже применяли для СППЗУ. Таким образом, совокупность входов логического элемента И представляют в коммутационном поле горизонтальной линией, которая пересекает все вертикальные линии, соответствующие входным термам или термам обратной связи, прямым или инвертированным.

Каждый вход, соединенный с входным термом (иначе говоря, неразрушенная перемычка), обозначен крестом или точкой в месте пересечения соответствующих вертикальной и горизонтальной линий.

В действительности наиболее простые ПЛМ часто располагают восемью входами и восемью выходами, и это уже дает большие возможности. Достаточно использовать корпуса только с 20 выводами – «двадцатую серию» ПЛМ. Они были наиболее широко распространены до появления «двадцать четвертой серии» с двадцатью четырьмя выводами и соответственно с большим количеством входов и выходов.

С увеличением числа входов и выходов схема усложняется: на рис. 3.3 показано, как выглядит матрица ПЛМ с четырьмя входами, четырьмя выходами и шестнадцатью термами произведений, но это далеко не предел.

САМЫЕ ПОПУЛЯРНЫЕ ПЛМ

Архитектура, которую мы только что определили, допускает многочисленные варианты. И это очень удобно, поскольку не каждая ПЛМ позволяет решить любую проблему с логическими схемами.

Когда есть потребность в большом количестве входов и выходов, можно объединить несколько корпусов ПЛМ, сохраняя при этом выигрыш в количестве корпусов до четырех или пяти раз относительно использования стандартных логических схем серий 74 или 4000. Предел сложности зависит от конечного числа термов произведений, подаваемых на входы каждого вентиля ИЛИ.

По сравнению с ПЛМ, ППЛМ (программируемые пользователем логические матрицы – Field Programmable Logic Array, FPLA) обладают большей гибкостью в организации схем, так как имеют второе программируемое коммутационное поле – между вентилями И и вентилями ИЛИ. При этом соответственно возрастает сложность и увеличивается время разработки приложений.

Подобную гибкость можно получить и от обычных ПЛМ, выпуская микросхемы с различными вариантами соединений между модулями с использованием одной общей архитектуры.

Обозначение каждой ПЛМ помимо информации второстепенного значения содержит корень, информирующий о внутренней организации радиоэлемента.

ПЛМ 16L8, например, имеет матрицу с 16 входами (либо 32 вертикали, если учитывать инверсные) и восемью выходами с инверторами (с активным низким уровнем – active Low).

Такие же возможности имеет и ПЛМ 16R8, но перед ее выходами включены триггеры (Registered).

Широко распространены промежуточные конфигурации, которые различаются количеством выходов, снабженных буферами или триггерами. В ПЛМ 16R4 встроены, например, только четыре триггера, в то время как в ПЛМ 16R6 их сделано шесть и, следовательно, есть только два обычных буфера.

Буква «H» означает, что комбинаторные выходы сделаны без инверсии (с активным высоким уровнем – active High), буква «X» – что ПЛМ снабжена выходными ячейками с логическими элементами Исключающее ИЛИ (eXclusive OR), а «P» соответствует ПЛМ, в которых программируется тип выходного буфера – повторитель или инвертор.

Чтобы изучить основы работы с ПЛМ, предпочтительно ограничиться микросхемами 16L8 и 16R8, а также, по необходимости, 16R6 и 16R4. Это наиболее характерные микросхемы, которые лучше всего подходят для образовательных целей.

На рис. 3.4 приведены цоколевки всех четырех упомянутых выше микросхем в режиме «использование», которые полностью отличаются от цоколевки, применяемой при программировании этих приборов.

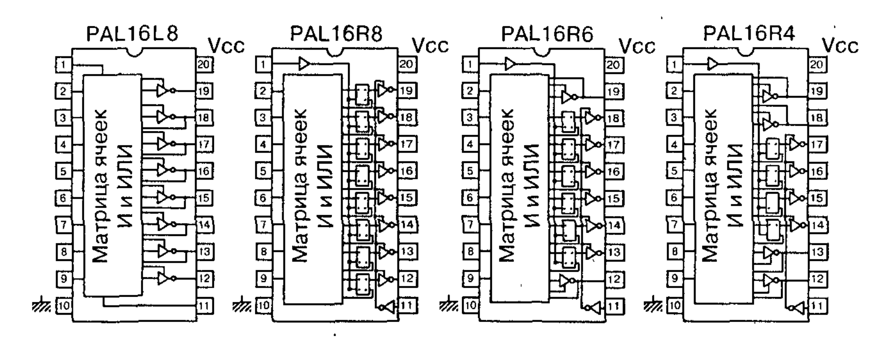


Рис 3.4. Цоколевки наиболее распространенных ПЛМ

ПЛМ 16L8

Рис. 3.5 полностью воспроизводит внутреннюю схему ПЛМ 16L8 (без привязки к тому или иному изготовителю) с обозначениями, оговоренными ранее. Но для удобства чтения кресты, соответствующие неразрушенным плавким перемычкам, не использовались. (Реализуемая логическая функция определяется теми плавкими перемычками, которые оставляют нетронутыми, поэтому разумно кресты добавлять, а не стирать.)

Данная ПЛМ имеет десять входных выводов (с 1 по 9 плюс 11), причем 11 часто служит входом разрешения для выходных буферов, хотя это и не обязательно.

Из десяти входных сигналов образуются 10 прямых и 10 инверсных, которые подключаются к 20 вертикальным линиям программируемой матрицы (коммутационного поля), содержащей 32 такие линии (это линии 0, 1, 2, 3, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 30, 31 – см. рис. 3.5).

На 12 оставшихся линий подаются сигналы обратной связи от шести из восьми выходных буферов, также и прямые, и инверсные. Каждый из восьми выходов (выводы с 12 по 19), оснащенных буферами с третьим состоянием, можно переключить в неактивное (третье) состояние навсегда, но это происходит и в зависимости от некоторых состояний на входах. В таких случаях шесть из выходных выводов (выводы с 13 по 18) могут применяться как дополнительные входы, что позволяет довести максимальное количество одновременно используемых входов до 16, но при этом, естественно, можно получить не более двух выходов.

Каждый из восьми логических элементов ИЛИ, работающих на выходы, имеет семь входов и способен обрабатывать семь термов произведений от пятидесяти шести 32-входовых логических элементов И. Таким образом, остаются восемь термов произведений из 64 доступных (0, 8, 16, 24, 32, 40, 48 и 56). Они выделены для независимого управления режимом восьми выходных буферов с третьим состоянием, что дает возможность исключительно гибко организовать систему ввода/вывода разрабатываемой микросхемы.

На этом этапе нашего знакомства с интегральными микросхемами ПЛМ полезно ввести два основополагающих правила:

- любое сочетание сигнала и его дополнения в терме произведения переводит последний в неизменное состояние лог. 0 (на выходе всегда низкий уровень). Это, в частности, имеет место, если все плавкие перемычки целы, например когда ПЛМ еще не программировалась;
- если же все перемычки линии разрушены, соответствующий терм произведения постоянно находится в состоянии лог. 1 (на выходе всегда высокий уровень).

Следовательно, для того чтобы зафиксировать выходной буфер в третьем состоянии (с высоким выходным сопротивлением), достаточно оставить нетронутыми все перемычки линии, управляющей его

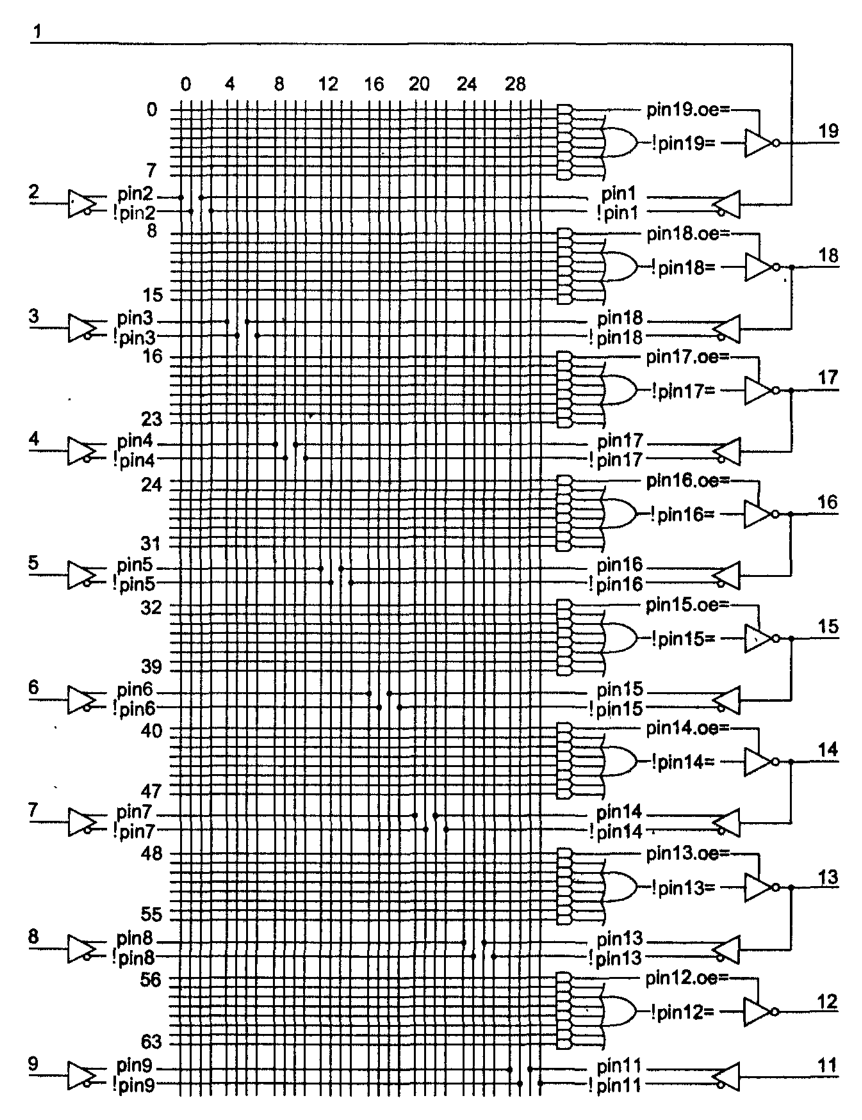


Рис. 3.5. Внутренняя схема ПЛМ 16L8

входом разрешения. Напротив, чтобы навсегда превратить этот буфер в простой инвертор или повторитель, надо полностью разрушить все перемычки той же линии.

ПЛМ 16R8

На рис. 3.6 показана внутренняя схема ПЛМ 16R8. Отметим наличие только восьми входов и восьми выходов, каждый из которых оснащен D-триггером и следующим за ним буфером с третьим состоянием.

В отличие от ПЛМ 16L8 восемь буферов управляются здесь по одной линии разрешения, подключенной к выводу 11. Другое отличие

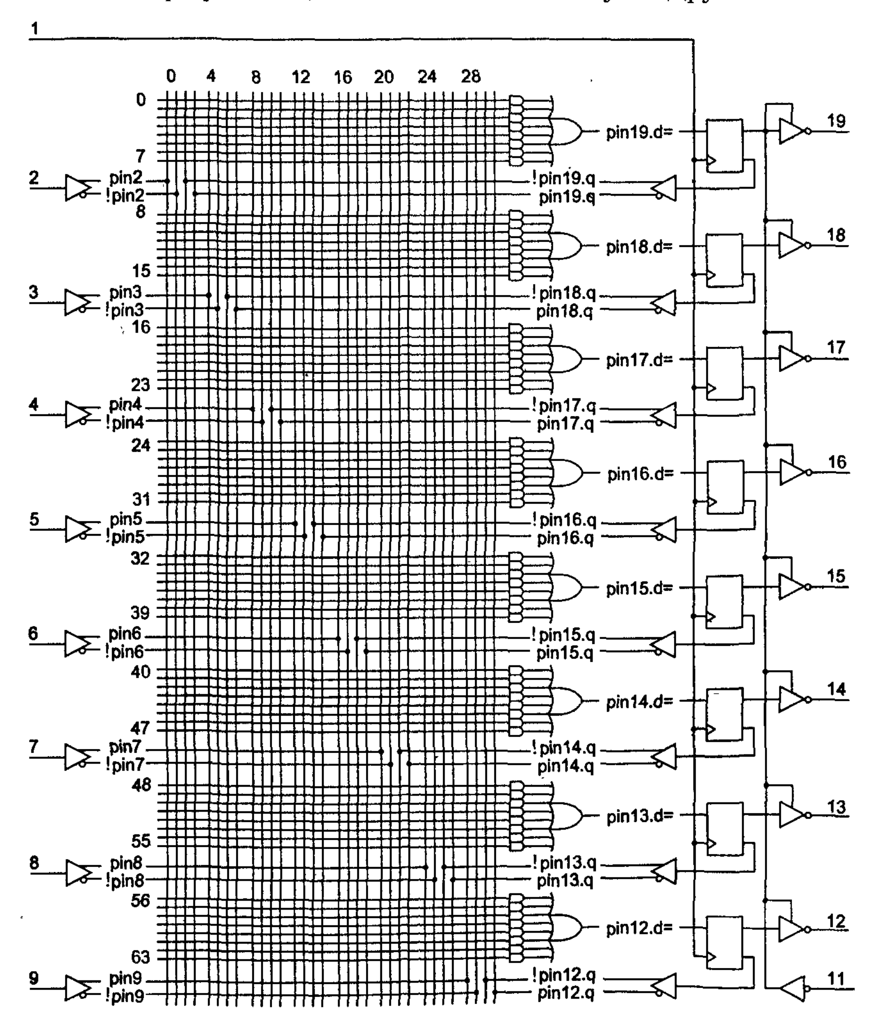


Рис. 3.6. Внутренняя схема ПЛМ 16R8

заключается в том, что выходы не имеют цепи обратной связи, поэтому не могут использоваться в качестве дополнительных входов. Однако состояния выходов всех восьми триггеров передаются с помощью буферов с парафазными выходами обратно в программируемую матрицу, следовательно, в ней остаются те же 32 вертикальные линии.

Как известно, обычный D-триггер можно преобразовать в Т-, RS- или даже в JK-триггер, если разместить на его входах соответствующие комбинаторные схемы, поэтому конфигурация микросхемы ПЛМ 16R8 построена так, чтобы упростить реализацию всевозможных счетчиков, а также сдвиговых и другие регистров, в состав которых.входит до восьми триггеров.

Входы синхронизации восьми триггеров соединены вместе на линии, подключенной к выводу 1. Таким образом, разрабатывать можно только синхронные последовательные схемы (все стандартные счетчики – синхронные, но синтезировать синхронные схемы несколько сложнее, чем асинхронные).

ПЛМ 16R4 и 16R6

Микросхемы 16R4 и 16R6 являются производными одновременно от ПЛМ 16L8 и 16R8. С их помощью можно решать еще более разнообразные задачи, хотя базовый принцип архитектуры остался прежним.

УНИВЕРСАЛЬНЫЕ ПЛМ И GAL

Итак, мы рассмотрели четыре различных типа ПЛМ, которые в настоящее время наиболее распространены. Мы увидели, что определенный недостаток гибкости архитектуры ПЛМ можно скомпенсировать только очень обстоятельным выбором типа ПЛМ для каждого конкретного применения. Но запасаться множеством типов и разновидностей микросхем ПЛМ совсем не обязательно.

В течение долгого времени многие компании-производители разрабатывали концепцию «универсальных» ПЛМ, которые благодаря дополнительным плавким перемычкам могли бы программироваться таким образом, чтобы реализовывать архитектуру различных стандартных ПЛМ.

Микросхема GAL22V10 – несомненно наиболее популярная из программируемых логических схем после ПЛМ с 20 выводами, которые она, похоже, начинает все больше и больше вытеснять. Термин «GAL» (Generic Array of Logic – логическая матрица общего применения,

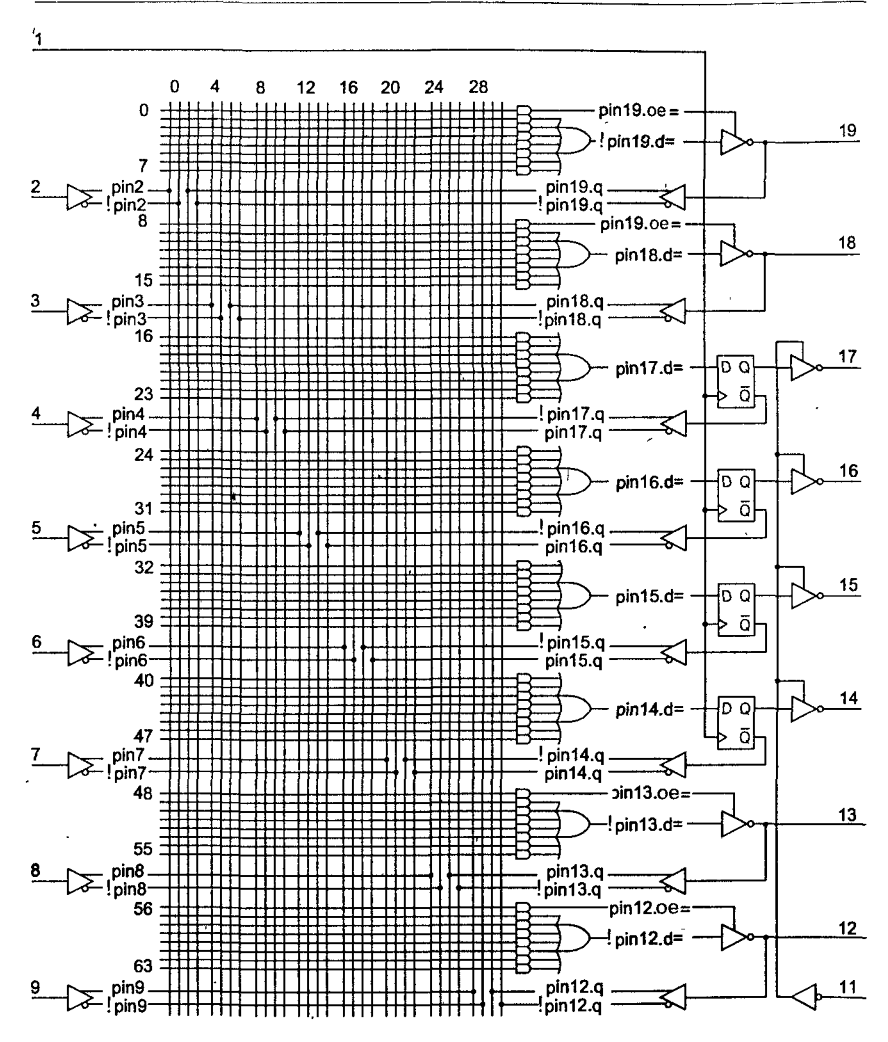


Рис. 3.7. Внутренняя схема ПЛМ 16R4

универсальная логическая матрица) – это зарегистрированная торговая марка компании Lattice, но он уже давно используется другими фирмами, которые производят эти радиоэлементы. Некоторые предпочли придумать синонимы, например компания AMD назвала аналогичнее семейство PALCE, а компания ICT выбрала аббревиатуру PEEL.

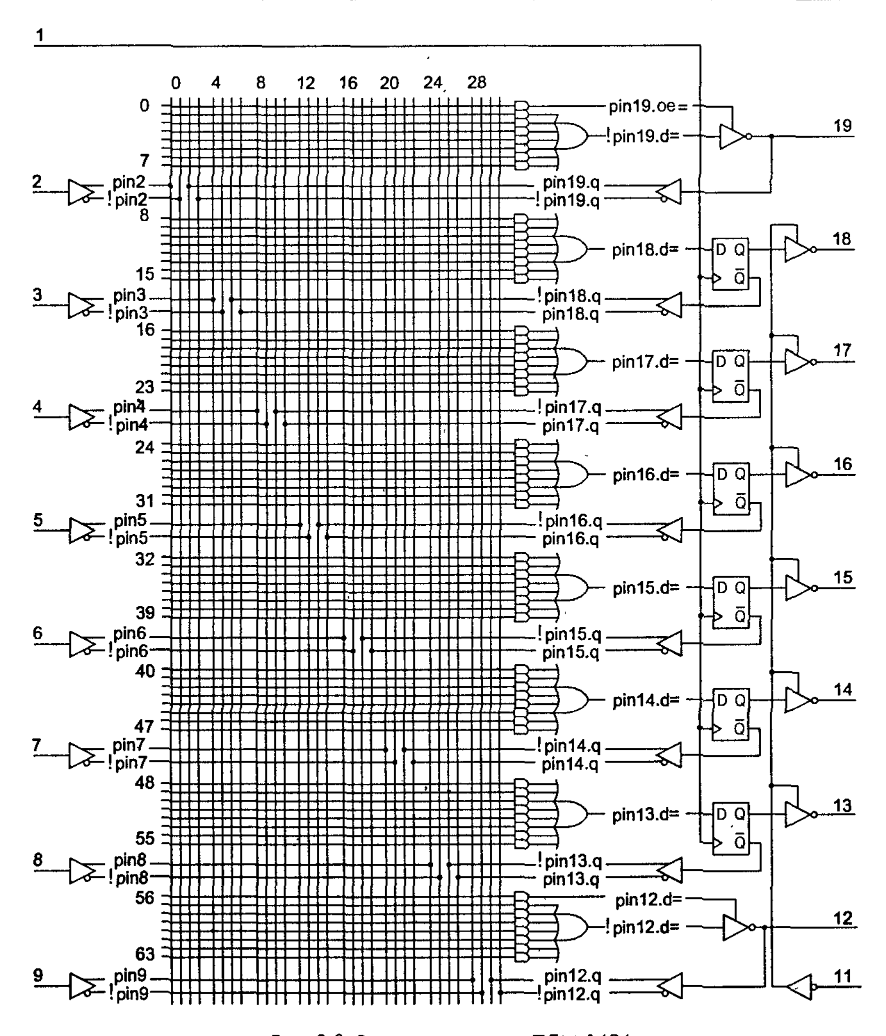


Рис. 3.8. Внутренняя схема ПЛМ 16R6

Принцип GAL – это вариант архитектуры ПЛМ, главная особенность которого состоит в том, что все выходные ячейки здесь тоже программируемые. Таким образом, там, где в стандартной ПЛМ (например, в 16R6) находится триггер или буфер, в микросхеме GAL расположена макроячейка, которая способна быть и триггером, и буфером, а выход макроячейки может быть и прямым, и инверсным. Очевидно, что

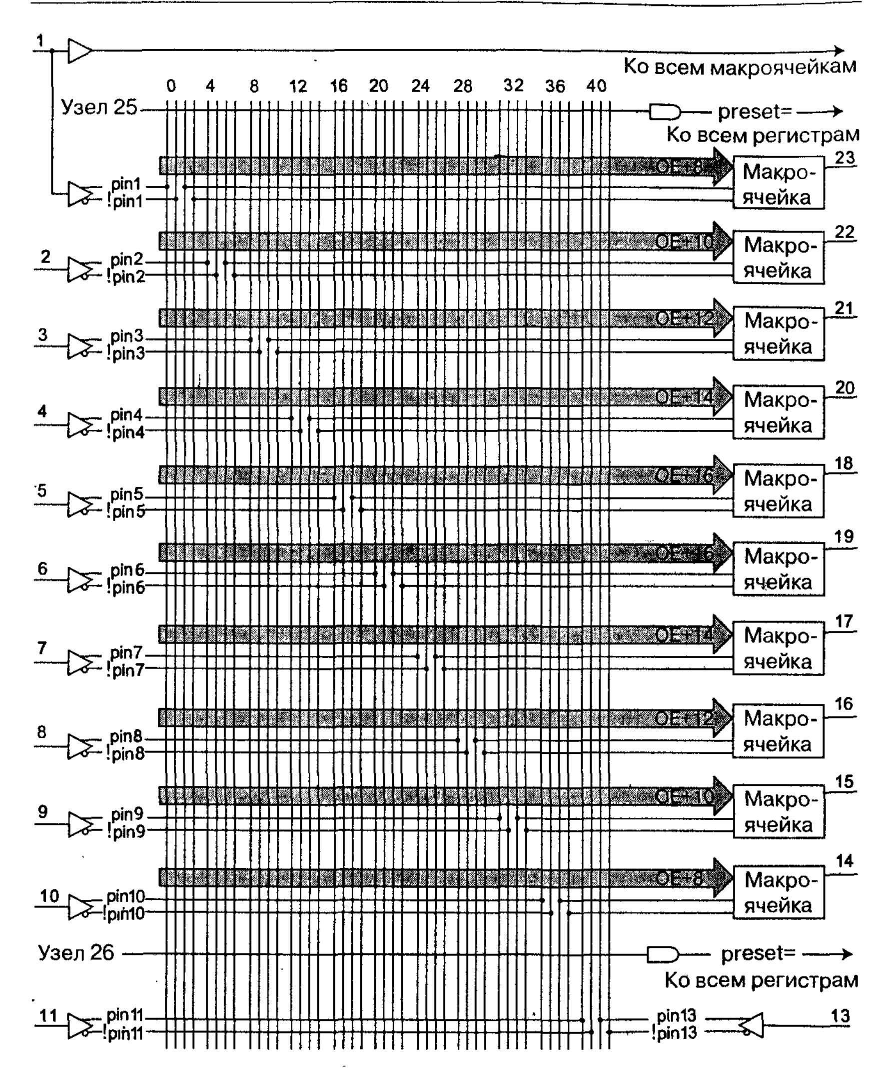


Рис. 3.9. Программируемая коммутационная матрица микросхемы GAL22V10

это более удобный вариант организации. Поскольку в микросхеме GAL22V10 имеется 10 макроячеек и 22 возможных входа, она помогает решать достаточно сложные задачи и строить развитые логические схемы, так как эта микросхема эквивалентна примерно 500 вентилям[1].

[1] Обычно под словом «вентиль» подразумеваются двухвходовые элементы И-НЕ и ИЛИ-НЕ. – *Прим. ред.*

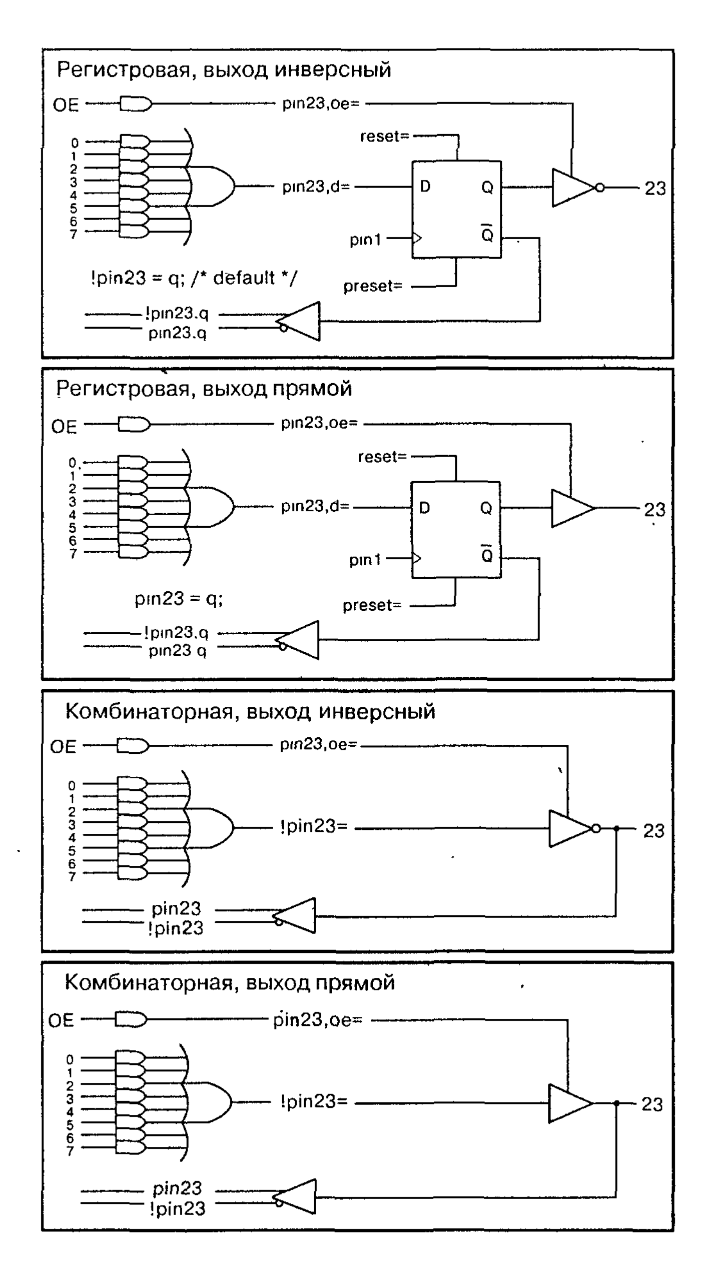


Рис. 3.10. Возможные конфигурации макроячеек микросхемы GAL22V10

На рис. 3.9 представлена схема программируемой матрицы, служащей для выполнения нужных соединений между всеми макроячейками, а на рис. 3.10 приведены четыре конфигурации, которые можно задать для каждой макроячейки при программировании.

Скопированные из руководства к логическому компилятору Prologic – программы, распространяемой компанией Texas Instruments (см. главу 6), – эти два документа представляют обозначения сигналов, которые надо использовать при подготовке исходного файла из булевых уравнений, таблиц истинности или диаграмм состояний.

Конфигурация каждой макроячейки согласно той или другой из возможных схем получается автоматически, с учетом соответствующего обозначения использованнных сигналов (pin23.d=, pin23= или !pin23=) и объявления, в случае триггера, прямой или инверсный нужен сигнал обратной связи (pin23=q; или !pin23=q;).

После компиляции полностью подготовленного, проверенного и отмоделированного исходного кода программа Prologic генерирует файл, предназначенный для программатора, в котором определено, какие перемычки следует уничтожить, а какие оставить как в коммутационной матрице, так и в макроячейках микросхемы GAL.

ЧТО СОДЕРЖАТ ПЛМ И GAL

Итак, мы убедились в том, что даже с «второсортными» ПЛМ, которые мы рассмотрели выше, можно реализовывать относительно сложные цифровые логические схемы (комбинаторные и/или последовательные) на весьма малом количестве корпусов, часто даже на одном. Для этого надо определить список необходимых внутренних соединений, то есть написать соответствующую программу, а затем перенести полученные данные в радиоэлемент с помощью программатора.

Радиолюбитель, привыкший к созданию цифровых систем, очевидно, мог бы выполнить программирование матрицы просто расставляя кресты или точки на копии, сделанной с исходных схем, представленных на рис. 3.5–3.9. Но более удобным будет составить таблицу истинности или диаграмму состояний или просто написать систему булевых уравнений.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАЗРАБОТКИ

Сегодня практически никто не создает цифровые схемы на основе ПЛМ или GAL без специальных программ, называемых логическими

компиляторами, которые предназначены для функционирования на совместимых ПК.

К числу известных программ относятся Palasm (наиболее старый, разработка компаний AMD/MMI), Abel (компания Data I/O), Cupl (компания Assisted Technology), Plan (корпорация National Semiconductor) и Prologic (фирма Inlab, пакет распространяется в виде ограниченной версии компанией Texas Instruments). Регулярно появляются и новые, все более мощные программы.

Как правило, компиляторы, способные поддерживать радиоэлементы многих производителей и подбирать конкретную модель в зависимости от решаемой задачи, стоят дорого. К счастью, некоторые производители ПЛМ и GAL бесплатно предлагают версии таких программ, предназначенные исключительно для их собственных радиоэлементов. (Рекламные цели подобных действий вполне понятны: ориентация радиолюбителя на приобретение продукции конкретной фирмы.)

Роль логического компилятора заключается в том, чтобы перевести булевы уравнения, таблицы истинности или диаграммы состояний (и даже принципиальные схемы) в так называемый файл JEDEC – закодированный в стандартизированной форме список плавких перемычек, которые следует уничтожить.

Ниже приводится пример очень простой схемы, реализованный с помощью программы Prologic – это логический элемент И-НЕ с тремя входами на ПЛМ 16R4. Файл называется NAND3.PLD:

```
include p16r4;
!pin19 = pin2 & pin3 & pin4;
pin 19.oe = 1;
test_vectors {
        pin2    pin3    pin4    !pin19 ;
        0       0       0       L      ;
        0       0       1       L      ,
        0       1       0       L      ;
        0       1       1       L      ;
        1       0       0       L      ;
        1       0       1       L      ;
        1       1       0       L      ;
        1       1       1       H      ;
}
```

Строка Include служит для объявления типа ПЛМ, чтобы компилятор мог применять нужную базу данных во время выполнения работы. Далее следует уравнение, составленное в виде строки текста с использованием условных знаков: восклицательный знак – для инверсии, амперсанд – для логической операции И и т.д.

Одна независимая строка служит для того, чтобы установить постоянный высокий уровень на линии разрешения буфера, обслуживающего вывод 19. Отметим, что обозначение этого сигнала взято из рис. 3.7.

Наконец, предусмотрена факультативная часть – Test vectors. Она предназначена для задания состояний, ожидаемых от программируемого радиоэлемента, без учета предварительно составленных уравнений. За соответствие одного другому, очевидно, отвечает только программист.

По окончании компиляции можно выполнить логическое моделирование, чтобы проверить, даст ли программирование радиоэлемента ожидаемые результаты. Это можно сделать, даже не располагая программатором.

Текст подобной программы создается в любом текстовом редакторе и сохраняется на дискете в виде файла с расширением .pld. В нашем случае этот файл будет называться nand3.pld. Тогда достаточно запустить логический компилятор, набрав на клавиатуре LC NAND3, чтобы через несколько мгновений получить файл JEDEC с именем nand3.jed, пример которого приведен ниже:

```
proLogic Compiler
Texas Instruments V1.97
Copyright (C) 1989 INLAB, Inc.

P16r4 revision 89.2.11

*N_csidp16r4
*QP20
*QV8
*QF2048
*F0
*L0000 11111111111111111111111111111111
*L0032 01110111011111111111111111111111
*C07E6
*V1 N000NNNNNNNNNNNNNNHN
*V2 N001NNNNNNNNNNNNNNHN
*V3 N010NNNNNNNNNNNNNNHN
*V4 N011NNNNNNNNNNNNNNHN
*V5 N100NNNNNNNNNNNNNNHN
*V6 N101NNNNNNNNNNNNNNHN
*V7 N110NNNNNHNNNNNNNNHN
*V8 N111NNNNNNNNNNNNNNLN
*V6D0C
```

Здесь представлен кодированный список перемычек, которые следует уничтожить (1) или оставить целыми (0), а также набор «тестовых векторов», позволяющих контролировать программируемую схему (если они заданы в исходном тексте).

Чтобы приступить к моделированию, достаточно набрать на клавиатуре LS NAND3 и ожидать результатов.

Всякий хороший программатор ПЛМ может работать с файлами JEDEC и в соответствии с ними запрограммировать чистую ПЛМ. Для этого файл содержит некоторую дополнительную информацию: тип ПЛМ (здесь 16R4), число выводов (здесь 20), количество тестовых векторов (здесь 8) и общее количество перемычек в радиоэлементе (здесь 2048).

Файл JEDEC предназначен для управления оборудованием, и интерпретировать его содержание непросто. Поэтому программа Prologic компилирует дополнительный файл листинга, называемый nand3.lst, который очень нагляден и удобен для анализа:

```
Copyright (C) 1989 INLAB, Inc.
Fuse Plot

P16r4 revision 88.2.11

11 1111 1111 2222 2222 2233
0123 4567 8901 2345 6789 0123 4567 8901

0 - - - - - - - - -    OE
1 X- X- X- - - - - - - +
2 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +
3 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +      !pin 19
4 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +
5 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +
6 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +
7 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +

8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  OE
9 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX  +
10 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX +
11 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX +      !pin 18

(.....)

61 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX +
62 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX +
63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX +
```

```
       |    |    |    |    |    |    |    |    |    |
|      |    |    |    |    |
pin2   |    3    |    4    |    5    |    6    |    7
|      8    |    9    |
       |    |    |    |    |    |    |    |
       pin1 18   17   16   15   14   13   11

Legend:

X : Cell intact      (JEDEC 0)
- : Cell programmed  (JEDEC 1)
X- : True input term, Complement register term
-X : Complement input term. True register term
XX : Any XX pair in a product term yields product term LOW.
: No input term (don't care). A product term comprised entirely of - yields
product term HIGH.
```

Карта перемычек здесь представлена очень ясно и подробно и сопровождается несколькими важными комментариями на английском языке.

Следовательно, файл JEDEC может служить непосредственно для программирования микросхем ПЛМ 16R4, но его разрешается конвертировать для программирования универсальной ПЛМ или для микросхем GAL.

Ниже показан пример такого преобразования для программирования микросхемы PEEL 18CV8 компании ICT (с 2696 перемычками), а затем приведен результат еще одного преобразования – для микросхемы EP320 компании ALTERA (с 2916 перемычками).

```
 JEDEC PEEL file Translated from: PAL16R4    Wed 1-2-1980   0:12:34
      nand3.JED
*DF PEEL
*DD 18CV8
*DM ICT
*QP20
*QF2696
*F0
 *
N Output Pin 19  *
L0000 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0036 1111 0111 0111 0111 1111 1111 1111 1111 1111  *
L0072 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0108 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0144 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0180 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0216 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0252 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
```

```
N Output Pin 18  *
L0288 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
L0324 0000 0000 0000 0000 0000 0000 0000 0000 0000  *
(...)

proLogic Compiler
Texas Instruments V1.97
Copyright (C) 1989 INLAB, Inc.

EP320
P16r4 revision 89.2.11

*N
PAL2EPLD Version 1 1 - Nov 18, 1986
EP320 JEDEC from conversion of PAL JEDEC*
N_csidp16r4
*QP20
*QV8
*QF2916
*L0
11101010111111111111111111111111111
11000000000000000011000000000000000
11000000000000000011000000000000000
11000000000000000011000000000000000
11000000000000000011000000000000000
11000000000000000011000000000000000
(.....)
```

Не будем останавливаться на внутренних различиях, существующих между этими радиоэлементами, которые, однако, будучи один раз запрограммированными, становятся идентичными на уровне выводов. Но запомним, что синтаксис файлов JEDEC относительно гибок и допускает многочисленные варианты.

В качестве примера приведем результат обработки информации, выполненной совершенно другой программой – утилитой JED2EQN, предоставленной компанией Altera: восстановление логических уравнений из файла JEDEC.

```
proLogic Compiler
Texas Instruments V1.97
Copyright (C) 1989 INLAB, Inc.

P16r4 revision 89.2.11
JED2EQN version. 1.0 beta 7 processed: Wed Jan 02 00:39:38 1980
PART: auto
INPUTS:
```

```
        pin1, pin2, pin3, pin4, pin11
OUTPUTS:
        pin14, pin15, pin16, pin17, pin19
NETWORK:
Pin1  = INP(pin1)
pin2  = INP(pin2)
pin3  = INP(pin3)
pin4  = INP(pin4)
pin11 = INP(pin11)
pin14 = RONF(din14. clk1, GND, GND, oen11)
pin15 = RONF(din15, clk1, GND, GND, oen11)
pin16 = RONF(din16, clk1, GND, GND, oen11)
pin17 = RONF(din17, clk1, GND, GND, oen11)
pin19 = CONP(din19. VCC)

EQUATIONS:
clk1        =      pin1
                   ;
oen11       =      !pin11
                   ;
!din14      =      GND
                   ;
!din15      =      GND
                   ;
!din16      =      GND
                   ;
!din17      =      GND
                   ;
!din19      =      pin2   &    pin3   &    pin4
                   ;
END$
```

Разумеется, цель этого преобразования – произвести исходный файл, пригодный для обработки собственным логическим компилятором этого производителя, однако полученные результаты можно использовать и для других целей, например для проверки результатов компиляции.

Уточним, что если файл JEDEC и может быть при необходимости отредактирован, модифицирован или просто написан заново при помощи простого текстового редактора, то для этого все равно существуют специализированные утилиты.

На демонстрационной дискете программаторов Sprint, в частности, содержится превосходный «редактор перемычек». Все их изменения отображаются на «карте перемычек» в явном виде, но сохраняются на дискете уже в виде настоящего файла JEDEC.

Занимающее обычно несколько дискет, это исключительно удобное программное средство приведено на сайте www.dmk.ru.

ПРОГРАММАТОРЫ

Созданный вручную или полученный от компилятора, файл JEDEC может быть использован с любым программатором микросхем ПЛМ или GAL для программирования чистого радиоэлемента и затем для проверки результатов операции.

Если все ПЛМ с одинаковой внутренней архитектурой совместимы между собой на уровне файла JEDEC, то алгоритмы программирования у разных изготовителей сильно отличаются друг от друга.

Под алгоритмом программирования следует понимать порядок действий, которые нужно выполнить, чтобы разрушить данную перемычку: на какие выводы какие уровни напряжения надо подавать и как долго их удерживать.

Очевидно, что биполярные ПЛМ с металлическими перемычками и ПЛМ КМОП, использующие технологию СППЗУ, программируются разными способами. То же самое можно сказать и о PALC 16R8 компании Cypress и TICPAL 16R8 компании Texas Instruments: хотя они и выполнены по технологии КМОП, но программируются поразному. Поэтому промышленные программаторы снабжаются солидным ПО для персонального компьютера, которое организовано в большую базу данных и дает возможность настраивать прибор под каждый конкретный тип ПЛМ или GAL.

Наиболее простые программаторы поддерживают около десятка типов микросхем, в то время как более сложные распознают тысячи разновидностей микросхем, но и стоят в несколько раз дороже. Разумеется, периодически обновляемое ПО позволяет добавлять спецификации на вновь появляющиеся программируемые радиоэлементы.

Как правило, эти высокопроизводительные программаторы могут также программировать большое количество других радиоэлементов: СППЗУ, ППЗУ, ППЛИС, ЭСППЗУ, запоминающие устройства типа флэш и даже микроконтроллеры. Кроме того, некоторые программаторы СППЗУ могут снабжаться адаптерами для определенных типов ПЛМ, GAL или СПЛМ. Такого рода оборудование, очевидно, необходимо только тем, кто желает максимально подробно изучить все особенности программируемых радиоэлементов, то есть использовать для каждого проекта одну или несколько ПЛИС, лучше всего соответствующих поставленной задаче. Но в таком случае будет совершенно необходим еще и высококлассный логический компилятор.

Ясно, что такие затраты абсолютно неприемлемы для радиолюбителя или разработчика, желающего спокойно и обстоятельно заняться изучением программируемой логики.

Можно предложить следующий, более экономичный путь решения данной проблемы:

- выбрать несколько распространенных типов микросхем, доступных и недорогих. Стараться использовать только выбранные типы, даже если эффективность окажется неоптимальной для некоторых проектов;
- приобрести несколько разных микросхем, допускающих стирание информации, чтобы иметь возможность отлаживать их последовательно;
- достать бесплатный логический компилятор или начать разрабатывать карты программирования вручную;
- самостоятельно собрать простой программатор.

Однако необходимо располагать подробным алгоритмом программирования тех радиоэлементов, которые вышеупомянутый программатор должен будет поддерживать, и это несколько ограничивает ваш выбор.

БОЛЬШИЕ ПЛИС

Нет никакого сомнения, что будущее программируемой логики – это эквивалентные нескольким тысячам вентилей радиоэлементы, размещенные в одном корпусе для поверхностного монтажа с 44 выводами или более.

Вплоть до настоящего времени вхождение в эту область предполагало существенные затраты на оборудование, на программное обеспечение и, как ни странно, на обучение.

Современные методы программирования «на плате» – как isp, так и JTAG – произвели революцию, сделав создание специфических БИС доступным действительно для всех.

Напомним основные сокращения, используемые для определения степени интеграции цифровых микросхем.

Категория ИС (Small Scale Integration, SSI) – микросхемы малой степени интеграции, которые содержат несколько простых вентилей или один-два отдельных триггера. В любом случае степень интеграции практически не превышает эквивалента десятка вентилей.

Категория СИС (Medium Scale Integration, MSI) – микросхемы средней степени интеграции – объединяет базовые функции немного

более расширенных серий «74» или «4000»: дешифраторы, мультиплексоры, демультиплексоры и не слишком сложные триггерные схемы, такие как счетчики, регистры или защелки-фиксаторы. (Эквивалент нескольких десятков вентилей или, в крайнем случае, сотни.)

Категория БИС (Large Scale Integration, LSI) – большие интегральные микросхемы (большой степени интеграции) – стала выделяться как самостоятельная после того, как единственный кристалл начал выполнять сложную специфическую функцию (синтезатор частоты, дешифратор штрих-кодов, УСАПП, ИС телеуправления и т.д.). При этом достигаются плотности, соответствующие эквиваленту от нескольких сотен до нескольких тысяч вентилей.

Наконец, иногда употребляют термин «СБИС» (Very Large Scale Integration, VLSI) – сверхбольшие интегральные схемы, включающие в себя от нескольких тысяч до десятков тысяч вентилей: микросхема эквивалентна по сложности микропроцессору; впрочем, некоторые СБИС являются, по сути, микроконтроллерами, чьи ПЗУ (масочные) программируются специальным образом.

Программируемые логические радиоэлементы предлагают интересные альтернативы схемам на стандартных радиоэлементах этих категорий. Одним корпусом ПЛМ или GAL легко заменить до десятка кристаллов ИС или два-три кристалла СИС, а иногда и больше. Одна GAL22V10 эквивалентна примерно 500 вентилям, однако пытаться использовать на 100% эти ресурсы довольно сложно.

Но чтобы получить степень интеграции, соответствующую самым простым БИС, необходимо использовать несколько таких микросхем. Поэтому производители полупроводниковых радиоэлементов представили еще более совершенные программируемые структуры, приспособленные для самых масштабных проектов и систем.

Семейства MACH компании AMD или pLSI фирмы Lattice – самые яркие этому примеры, причем на некоторых микросхемах применяют наиболее совершенную концепцию программирования – «на плате», то есть практически без программатора.

Благодаря программному обеспечению для разработки и моделирования, исключительно легкому в использовании и иногда совсем недорогому, создание схем БИС частного применения теперь стало даже более простым, чем проектирование цифровых устройств на основе ПЛМ.

pLSI И ispLSI КОМПАНИИ LATTICE

Выпуск программируемых радиоэлементов pLSI помог компании Lattice (кстати, предложившей концепцию GAL) продвинуться на

одно из первых мест среди производителей программируемых логических схем с высокой степенью интеграции.

Нужно понимать, что возможности программируемой схемы LSI однозначно связаны и с логическими функциями, которые она содержит, и с количеством линий ввода-вывода, которыми она располагает, и с порядком организации и распределения соединений между этими ресурсами.

Семейства микросхем p/isp LSI1000, 2000 и 3000 охватывают широкую область применения, от 1000 до 14000 эквивалентных вентилей на кристалл, при этом используются корпуса, насчитывающие от 44 до 208 выводов.

С точки зрения чистой цифровой логики архитектура всех трех семейств основана на понятиях GLB (Generic Logic Block – логический блок общего назначения, или универсальный логический блок), GRP (Global Routing Pool, блок общей трассировки), ORP (Output Routing Pool, блок трассировки выходов) и I/O – ячейка ввода/вывода.

Каждый GLB своей архитектурой и возможностью перепрограммирования напоминает структуру микросхемы GAL. На рис. 3.11 показан логический блок, снабженный 18 входами и 4 выходами, программируемой матрицей из 20 логических элементов И (термов произведений) и 4 макроячейками, индивидуально программируемыми для выполнения логических функций (И, ИЛИ, Исключающее ИЛИ) или для работы в последовательных схемах (D-, T-, JK-триггеры, синхронные или асинхронные).

Отметим, что в радиоэлементах семейства 3000 используются сдвоенные GLB (см. рис. 3.12) с 24 входами, 40 термами произведений и 8 макроячейками.

Один простой GLB (серии 1000 и 2000) способен выполнять после обычного программирования до 90% обычных логических функций в 4-разрядных схемах. А для работы с байтами достаточно объединить два GLB.

На рис. 3.13 показаны различные способы использования макроячеек ввода/вывода вместе с участком полной схемы, который поможет лучше понять общую структуру GLB.

Если продолжать сравнения с семействами микросхем GAL, то простой GLB можно рассматривать, как своего рода «GAL 18V4», а два простых GLB или один сдвоенный GLB по своим возможностям приближаются к обычной GAL22V10.

Это родство «больших» ПЛИС с обычными ПЛМ и GAL очевидно и в семействе микросхем MACH 130 – радиоэлементов производства фирмы AMD, очень широко распространенных в последнее время.

На рис. 3.14 показана структура базового логического блока этого семейства, и внимательный анализ позволяет понять, почему один корпус MACH 130 может рассматриваться как объединение четырех воображаемых 26V16.

Эти микросхемы размещаются в корпусах PLCC с 84 выводами и по цоколевке полностью совместимы с микросхемами MACH 230 (эквивалент восьми 26V16), MACH 231 (восемь 32V16) и MACH 435 (восемь 33V16).

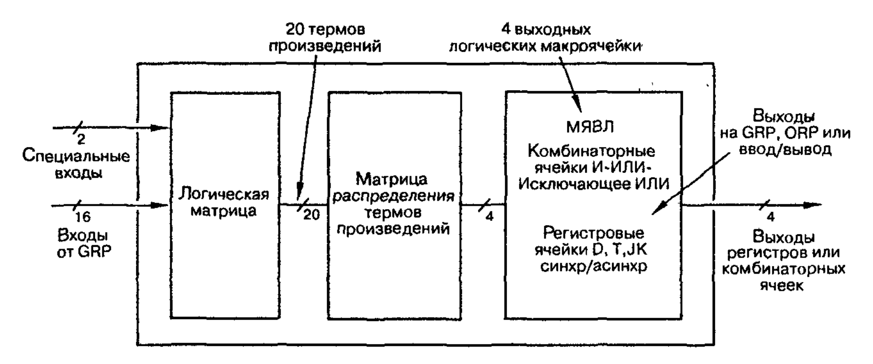


Рис. 3.11. Архитектура GLB

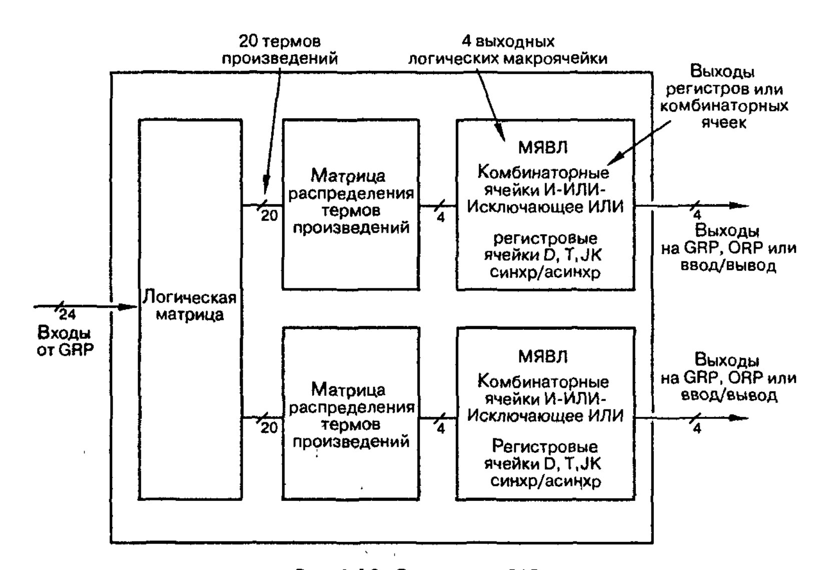


Рис. 3.12. Сдвоенная GLB

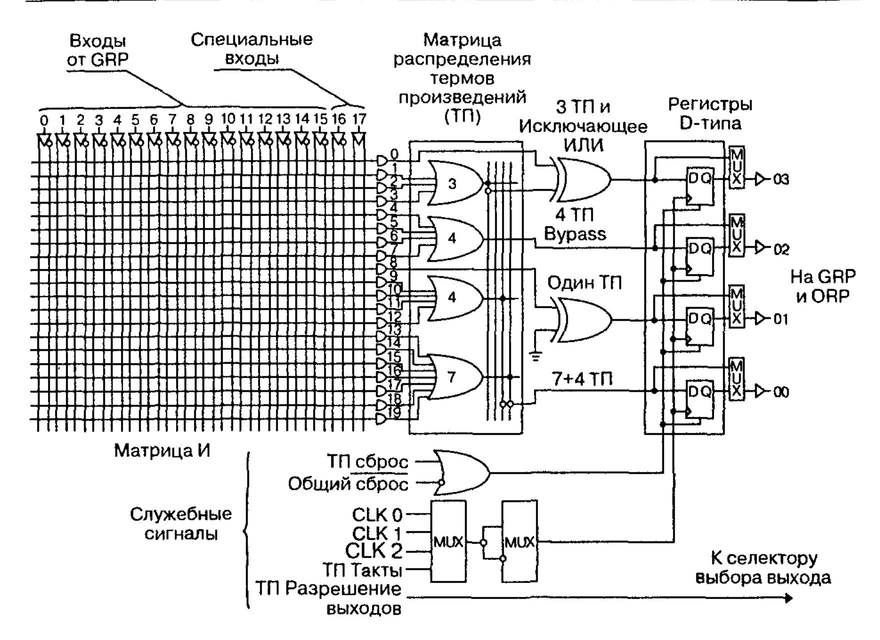


Рис. 3.13. Несколько примеров использования макроячеек

Микросхема MACH 355 производится в корпусе PQFP со 144 выводами и эквивалентна шести 33V16, но при этом имеет огромное преимущество, заключающееся в возможности программирования прямо «на плате», согласно протоколу JTAG, иначе говоря – без программатора.

Если провести небольшие доработки, то в некоторых случаях вполне допустимо заменить MACH 130 на MACH 355, в которую данные можно загружать, даже если она уже установлена на плату. Естественно, это открывает перед разработчиками еще более широкие перспективы.

У компании Lattice уже наиболее простые LSI серии 1000 (ispLSI 1016) содержат 16 GLB, хотя GLB – лишь один из четырех программируемых логических «слоев», которыми определяется конечная схема проектируемого устройства.

И действительно, перед GLB находится блок GRP, который, как указывает его название, используется для свободной трассировки (причем с постоянным временем распространения) взаимосвязей между всеми логическими ресурсами радиоэлемента, а также с его входными выводами.

После GLB расположены блоки ORP, предназначенные для трассировки соединений между выходами макроячеек и произвольно

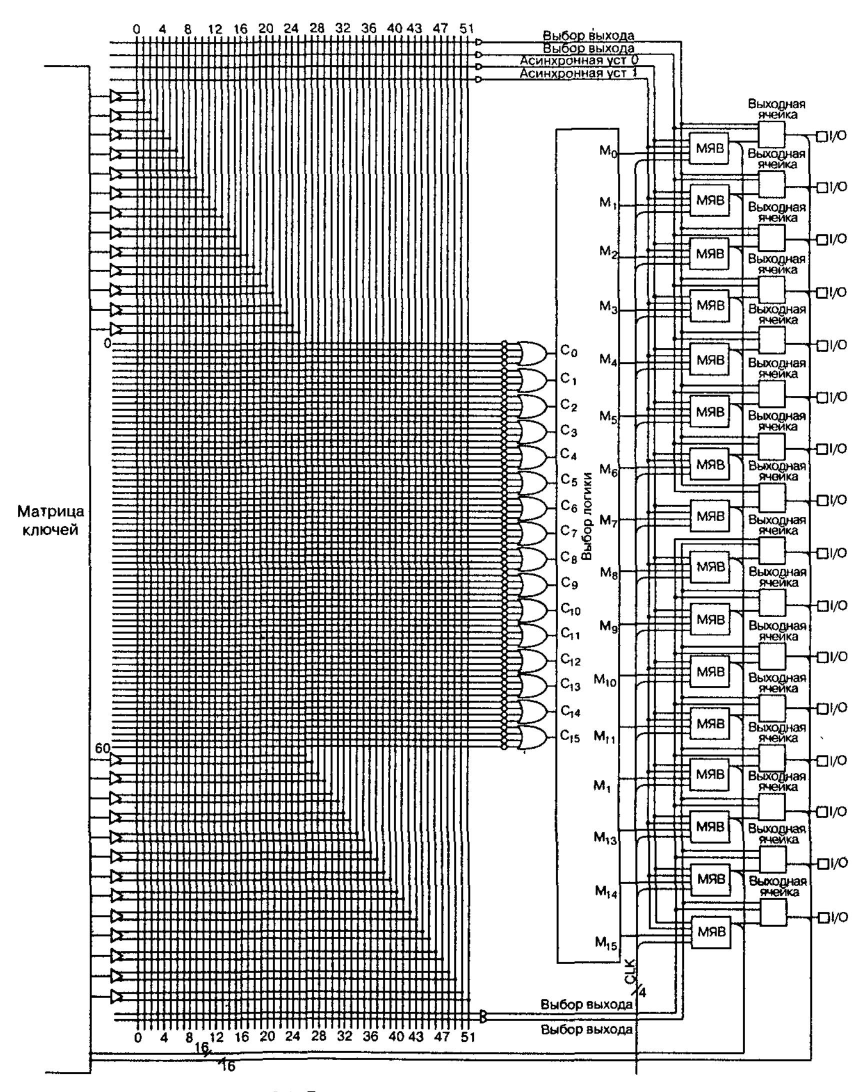


Рис. 3.14. Базовый логический блок MACH130

выбранными выводами кристалла, причем из расчета по одному ORP на четыре или восемь GLB. Некоторые выходы у каждого GLB могут также подключаться непосредственно к соответствующим ячейкам ввода/вывода, не проходя через ORP, при этом быстродействие схемы повышается.

Совокупность ячеек ввода/вывода – это целый программируемый слой. Принципиальная схема одной ячейки и расположение программируемых перемычек в ней показано на рис. 3.15.

Следует также отметить, что помимо этих универсальных линий ввода/вывода каждая микросхема LSI имеет несколько особых выводов: специальные входные (которые нельзя использовать в качестве выходных), входы сигналов синхронизации и, в случае необходимости, входы общего разрешения OE. Их количество и назначение могут незначительно отличаться от одного типа микросхем к другому, а также у разных изготовителей.

Все радиоэлементы из семейств p/ispLSI содержат некоторое количество мегаблоков, групп GLB, ячеек ввода/вывода, специальных входов и блоков ORP с общим термом произведения для сигнала разрешения OE. Их конкретные структуры немного отличаются от семейства к семейству.

Семейство 1000 состоит из четырех микросхем: p/ispLSI1016, 1024, 1032 и 1048, снабженных соответственно 2, 3, 4 и 6 мегаблоками (эквивалентными 2000, 4000, 6000 и 8000 вентилям).

Семейство 2000 начинается микросхемой с одним мегаблоком (p/isp LSI2032 – 1000 эквивалентных вентилей), а микросхемы семейства 3000 могут содержать до десяти (p/isp LSI3320 – 14000 эквивалентных вентилей).

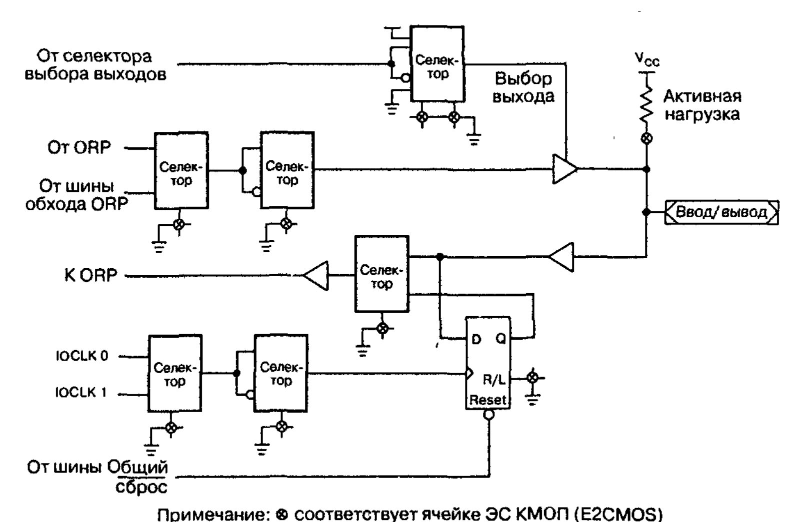


Рис. 3.15. Структура одной ячейки входа-выхода

Нет никакого сомнения в том, что сейчас микросхемы в корпусах с 44 выводами самые популярные (особенно те, которые допускают программирование «на плате»), поэтому первоочередной интерес следует проявлять к ispLSI2032 и 1016. Это именно те две микросхемы, которые поддерживаются программным обеспечением набора для начинающих (Starter Kit). В указанном наборе за весьма скромную цену предлагаются: специальный кабель для программирования, несколько образцов взаимно совместимых микросхем (среди которых один образец ispGAL22V10), необходимое программное обеспечение и комплект документации.

В табл. 3.1 сравниваются характеристики этих двух компонентов, а также некоторых других из семейства 1000.

Можно подсчитать, что на базе 1000 эквивалентных вентилей один корпус ispLSI2032 несравненно мощнее, чем микросхема GAL22V10, которая содержит, по крайней мере, 500 эквивалентных вентилей. Разница в мощности получается главным образом благодаря гибкости программирования внутренней структуры.

Программное обеспечение, которое содержит набор isp-Starter Kit, за исключением того, что оно ограничено только микросхемами 2032 и 1016, абсолютно идентично, включая и объем документации, пакету программ PDS, который поддерживает всю номенклатуру микросхем p/ispLSI.

Хотя затраты на приобретение такого набора для начинающих невелики, он позволяет разрабатывать схемы и устройства, значительно

Таблица 3.1. Основные микросхемы семейства ispLSI

	ispLSI2032	ispLSI1016	ispLSI1024	ispLSI1032	ispLSI1048
Макроячейки	32	64	96	128	192
GLB	8	16	24	32	48
Мегаблоки	1	2	3	4	6
Вентильный эквивалент	1000	2000	4000	6000	8000
Число контактов ввода/вывода	32	32	48	64	96
Число контактов ввода	2	4	6	8	10
Число ножек корпуса	44 (PLCC) 44 (TQFP)	44 (PLCC) 44 (TQFP) 44 (JLCC)	68 (PLCC) 68 (JLCC)	84 (PLCC) 84 (CPGA) 100 (TQFP)	120 (PQFP) 128 (PQFP)
Регистры	32	96	144	192	288
F max	135 МГц	110 МГц	90 МГц	90 МГц	80 МГц

превышающие по возможностям те, которые собраны на GAL или ПЛМ, пользуясь при этом всей мощью профессионального программного средства, снабженного удивительно удобным интерфейсом, поскольку программа рассчитана на работу в среде Windows. Кроме того, в состав столь недорогого программного продукта включена очень большая библиотека макросов и примеров.

Макросы, реализующие практически все логические функции, которые распространены в семействах цифровых ИС и СИС, можно извлечь непосредственно из объемного описания, прилагаемого к набору isp-Starter Kit. Одна из страниц этого описания представлена на рис. 3.16 в качестве примера.

Преобразователь двоичного кода в 7-сегментный BIN27 (который правильно работает даже с шестнадцатеричным кодом) можно расположить «в уголке» кристалла в двух GLB микросхемы, просто щелкнув кнопкой мыши по его имени в списке. Действительно, дешифратор имеет семь выходов и не может быть реализован на одном GLB, у которого есть только четыре выхода. Но большинство других макросов легко размещаются в одном GLB, и здесь нелишне напомнить, что даже ispLSI2032 имеет целых восемь GLB.

Любой проект с использованием ispLSI начинается с бумажной работы: с разделения мнемонической схемы или блок-схемы на несколько функций, каждая из которых должна находиться в одном или нескольких GLB. При этом становится очевидно, возможна ли интеграция данной схемы в микросхему выбранного типа. Затем можно начать работу с программным обеспечением PDS, в котором на экран монитора выводится упрощенное представление схемы используемого радиоэлемента.

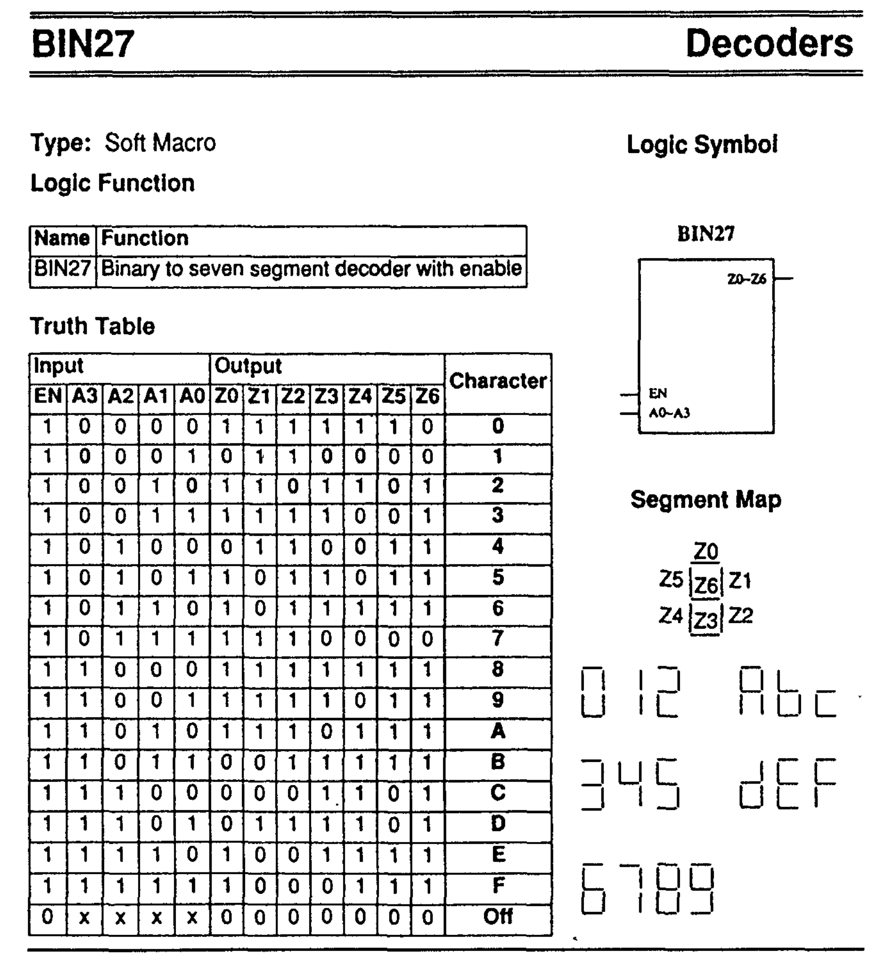
Много времени потребуется для заполнения необходимых GLB и ячеек ввода/вывода с помощью булевых уравнений, связанных с ними. Нужно, например, чтобы сигнал, входящий в GLB, носил то же имя, что и выходящий из ячейки входа/выхода или из другого GLB. И наоборот, выходящий из GLB сигнал должен соответствовать сигналу с тем же именем, входящим в ячейку ввода/вывода или в другой GLB.

Таким образом, для каждого GLB нужно вручную переименовать сигналы, входящие в состав уравнений используемого макроса, или непосредственно написать свои собственные уравнения, применяя имена действительно используемых сигналов и логические операторы, опознаваемые программой PDS. Это, очевидно, требует некоторого навыка.

Следует учесть и весь тот опыт, который вы приобретете во время подготовки на бумаге плана разделения схемы на функциональные

блоки, причем их входные и выходные сигналы должны быть полностью идентифицированы.

Так как в подобных процессах легко сделать неверный ход, в программу PDS включены мощные средства проверки и поиска ошибок. Сначала обрабатывается каждый GLB (предварительно обозначенный



BIN27 — Decoders

Type: Soft Macro

Logic Function

Name	Function
BIN27	Binary to seven segment decoder with enable

Truth Table

Input					Output							Character
EN	A3	A2	A1	A0	Z0	Z1	Z2	Z3	Z4	Z5	Z6	
1	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
1	0	0	1	0	1	1	0	1	1	0	1	2
1	0	0	1	1	1	1	1	1	0	0	1	3
1	0	1	0	0	0	1	1	0	0	1	1	4
1	0	1	0	1	1	0	1	1	0	1	1	5
1	0	1	1	0	1	0	1	1	1	1	1	6
1	0	1	1	1	1	1	1	0	0	0	0	7
1	1	0	0	0	1	1	1	1	1	1	1	8
1	1	0	0	1	1	1	1	1	0	1	1	9
1	1	0	1	0	1	1	1	0	1	1	1	A
1	1	0	1	1	0	0	1	1	1	1	1	B
1	1	1	0	0	0	0	0	1	1	0	1	C
1	1	1	0	1	0	1	1	1	1	0	1	D
1	1	1	1	0	1	0	0	1	1	1	1	E
1	1	1	1	1	1	0	0	0	1	1	1	F
0	x	x	x	x	0	0	0	0	0	0	0	Off

Logic Symbol

Segment Map

Equation Entry

BIN27 ([Z0..Z6],[A0..A3],EN)
BIN27_1([Z0..Z2],[A0..A3],EN)
BIN27_2([Z3..Z6],[A0..A3],EN)

Рис. 3.16. Описание макроса преобразователя кода BIN27

именем, облегчающим последующую его идентификацию), каждая ячейка ввода/вывода, а уже затем вся схема целиком. После того как все проверки будут завершены, можно запустить операцию, называемую трассировкой. Она заставит программное обеспечение определить пути, по которым внутри GRP и ORP должны следовать сигналы.

При этом можно выбрать вывод, выделенный для каждого входящего или выходящего сигнала. Такая операция позволяет значительно упростить разводку печатной платы, на которой будет размещен проектируемый радиоэлемент.

За исключением нескольких выводов, назначение которых определено изготовителем, разработчик сам выбирает, на какой вывод и на какой стороне корпуса PLCC следует вывести тот или иной сигнал.

В результате всех указанных операций можно получить листинг в формате LDF, представляющий собой текстовый файл, который в принципе мог бы быть написан непосредственно программистом, предварительно изучившим синтаксис этого специфического языка. Но гораздо проще использовать логический компилятор, такой как программа Prologic, в процессе программирования ПЛМ. Но в данном случае, очевидно, предпочтительнее просто соединить между собой несколько макросов, выбранных из каталога щелчком мыши.

На этой стадии остается только откомпилировать исходный текст с помощью того же программного обеспечения в файл JEDEC, совместимый с любым промышленным программатором, поддерживающим программирование микросхем pLSI и ispLSI.

Исполнение «isp» этих радиоэлементов делает возможным их программирование «на плате» посредством передачи данных для программирования по простой четырехпроводной шине – иначе говоря, вообще без дорогостоящих программаторов.

При серийном производстве это происходит, как правило, при прямом подключении к печатной плате иногда непосредственно в процессе выполнения автоматического теста JTAG.

Но в случае необходимости перепрограммирования.(это выполнимо и в стационарном, и в мобильном оборудовании) можно использовать главный процессор всей системы, передав данные либо на дискете, либо с помощью телефонного, радио- или иного модема.

Иными словами, при работе с микросхемами ispLSI можно одновременно пользоваться гибкостью программируемых матриц на базе ОЗУ (которые теряют хранящиеся в них данные после каждого отключения питания) и энергонезависимостью технологии ЭСКМОП (EECMOS).

Микропроцессор в современной электронике – «сердце» вычислительной техники, которая находит применение и в программных блоках стиральных машин, и в моторах автомобилей, и в устройствах управления лифтами и т.д.

Кроме того, микропроцессор – это, возможно, и лучший пример программируемой логики. Такой радиоэлемент совершенно ни к чему не пригоден без программы или программного обеспечения – подробного списка команд, которые следует выполнять. Более того, с точки зрения аппаратного обеспечения – это программа, размещенная в энергонезависимом запоминающем устройстве, которая определяет поведение системы: ее изменение может полностью изменить работу системы с микропроцессором.

СИСТЕМЫ С МИКРОПРОЦЕССОРОМ

На рис. 4.1 в общем виде показана классическая архитектура системы с микропроцессором. Собственно говоря, микропроцессор, или ЦПУ, окружен вспомогательными устройствами, необходимыми для его функционирования: это блоки питания, генератор тактовой частоты, запоминающие устройства и периферийные устройства ввода/вывода.

Тактовый сигнал, частота которого зависит от кварцевого резонатора, служит для синхронизации функционирования всей системы в целом и определяет скорость выполнения команд. Постоянные запоминающие устройства (ПЗУ, СППЗУ, ЭСППЗУ и даже устройства типа флэш) хранят выполняемую программу, каждая команда которой представлена в виде одного или нескольких последовательных слов (по 8 бит или больше). Оперативная память (ОЗУ) служит для временного хранения такой информации как промежуточных результатов сложных расчетов.

Периферийные устройства ввода/вывода применяются для получения и отправки информации. Они устанавливают связи с датчиками, реле, моторами автомобилей, сигнальными лампочками, а также с клавиатурой и экраном и т.д. Таким образом, микропроцессор – это интегральная микросхема, которая многочисленными линиями соединяется со всеми своими вспомогательными схемами.

В число этих линий входит, во-первых, шина данных, по которой передаются байты во время диалога с запоминающими или периферийными устройствами; она может содержать восемь линий для 8-разрядных микропроцессоров, 16 или 32 – для более мощных

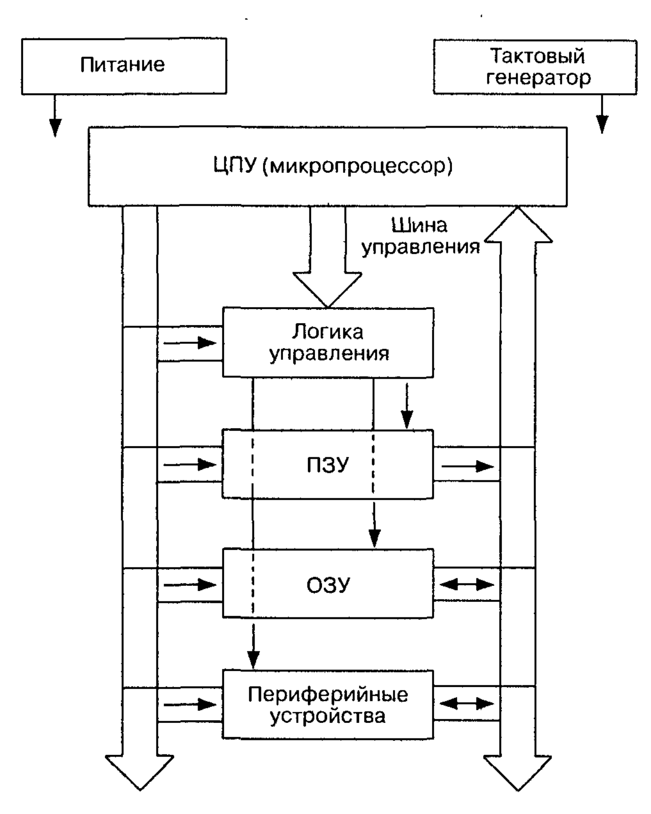


Рис. 4.1. Типовая архитектура системы с микропроцессором

микро-ЭВМ. Во-вторых, шина управления, определяющая, присутствует ли на шине данных слово, выдано ли оно самим ЦПУ или, напротив, предназначено для него, и проверяющая, передаются ли данные запоминающими или периферийными устройствами: это еще несколько линий, обычно четыре или пять.

Наконец шина адреса, которая служит для однозначного указания места в запоминающем устройстве или в пространстве портов ввода/вывода перед установлением связи с регистрами ЦПУ. Это еще 16 линий или больше. Зная, что каждый корпус запоминающего устройства имеет, как правило, 24 или 28 выводов и каждая схема ввода/вывода – порядка двадцати, можно представить всю сложность системы в целом.

МИКРОКОНТРОЛЛЕРЫ

Микроконтроллер, или однокристальная микро-ЭВМ, – это интегральная микросхема, объединяющая в одном и том же корпусе микропроцессор, память и устройства ввода/вывода. Архитектура такого

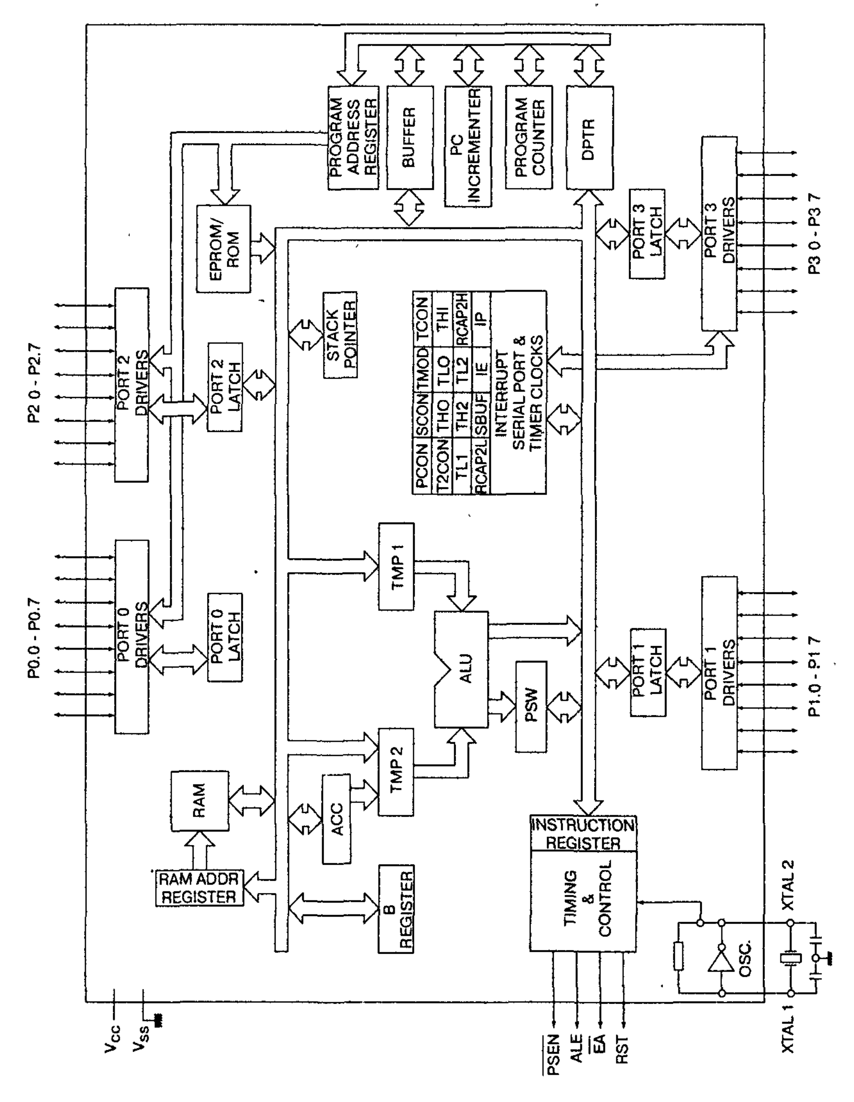


Рис. 4.2. Архитектура микроконтроллера 8051

радиоэлемента и его набор команд оптимизированы, как правило, для применения в устройствах и системах автоматики и других областях, где не требуется вычислительной мощности, но важны габариты и стоимость.

При этом достигается значительное упрощение конструкции и ощутимая миниатюризация конечного продукта, часто при незначительных затратах.

Число выводов корпуса зависит главным образом от количества необходимых линий ввода/вывода: классический корпус с 40 выводами вполне подходит для организации четырех групп по восемь линий ввода/вывода, в то время как некоторые более новые микроконтроллеры могут размещаться в маленьком корпусе всего с 8 выводами (микроконтроллеры PIC12Cxxx).

В качестве примера на рис. 4.2 приведена внутренняя структура микроконтроллера, принадлежащего к очень распространенному семейству 8051. Отметим, что это мощная микропроцессорная система, располагающая несколькими счетчиками-таймерами и портом последовательной связи на одном кристалле.

МИКРОКОНТРОЛЛЕРЫ СППЗУ И ОТР

Микроконтроллеры, снабженные внутренним масочным ПЗУ, используются только крупными компаниями и только тогда, когда надо изготавливать большие партии товаров в условиях серийного производства (*электробытовой техники, автомобилей и т.д.*).

Идеальным для разработки прототипов микроконтроллером является, очевидно, модель со встроенным ППЗУ, стираемым ультрафиолетовыми лучами (УФ СППЗУ – UV EPROM). Такие микросхемы можно запрограммировать, использовать, стереть с них информацию и при желании перепрограммировать заново так же легко, как и простое СППЗУ, характерный для которых керамический корпус с кварцевым окошком был использован для микроконтроллеров.

Не стоит забывать и о моделях ОП (OTP), размещенных в дешевых пластмассовых корпусах. Записанные в них данные стереть уже нельзя, следовательно, применяться они будут небольшими партиями, когда вся система уже полностью отлажена.

Микроконтроллер с СППЗУ – это действительно программируемый радиоэлемент: простая запись соответствующего программного обеспечения позволяет превратить стандартную радиодеталь в специфический радиоэлемент, способный выполнять чрезвычайно сложные функции.

Кроме того, в отличие от простых СППЗУ микроконтроллеры могут быть защищены от несанкционированного копирования данных и программного обеспечения из его запоминающего устройства.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И СИСТЕМЫ ПРОЕКТИРОВАНИЯ

Писать программу для микроконтроллера в машинных кодах – весьма скучное и тяжелое занятие. От нескольких десятков до нескольких сотен команд можно, конечно, написать и вручную, используя руководства и таблицы кодов, которые содержатся в технических описаниях радиоэлементов. Исправление ошибки или внесение изменения требуют значительных затрат времени на реорганизацию исполняемого кода. Но такого рода задача хорошо поддается компьютерной обработке. Теоретически любой компьютер или микро-ЭВМ можно приспособить для подготовки машинного кода, предназначенного для любого другого микропроцессора или микроконтроллера.

Программы, способные выполнять такие задачи, называются ассемблерами[1]. Вначале подготавливается текст исходной программы, написанный на языке ассемблера. Текст обрабатывается программой-ассемблером, и получается исполняемый модуль (в машинном коде), организованный в байты, которые можно записать в СППЗУ микроконтроллера.

Иногда в этой последовательности действий необходим промежуточный этап – объединение нескольких объектных программ, которые должны быть обработаны линкером, или редактором связей, для получения полноценной «зашивки». Кроме того, существуют программы-дизассемблеры, способные восстанавливать в виде списка мнемоники команд непосредственно из исполняемого кода.

Определенную проблему представляет и тестирование программы: мы слишком далеки от языка Basic, в котором простая команда Run позволяет провести испытание без риска «зависания» в случае ошибки.

Чтобы проверить программу микроконтроллера, надо в принципе «зашить» ее в СППЗУ, исполнить, чтобы отметить вероятные аномалии и ошибки, а затем соответственно изменить исходный текст, переассемблировать, стереть информацию в СППЗУ и перепрограммировать его новыми данными. Существуют, однако, различные методы, которые позволяют избежать подобных проблем и выиграть время.

[1] Если на ЭВМ одного типа готовятся коды для ЭВМ другого типа, то часто говорят о кросс-ассемблерах. – *Прим. ред.*

Сначала большинство программ-ассемблеров отмечают наиболее грубые ошибки программирования, позволяя исправлять их уже на ранней стадии. В случае если микро-ЭВМ, на которой работает программа-ассемблер, оснащена микропроцессором того же семейства, что и данный микроконтроллер, можно протестировать разработанную программу под управлением специальной программы-отладчика, с помощью которой устанавливают более серьезные ошибки, чтобы избежать серьезного «зависания» машины.

К сожалению, в большинстве случаев микропроцессор компьютера отличается от подготовленного к программированию микроконтроллера. Например, совместимый ПК оснащен 386 или 486 процессором, а программа разрабатывается для микроконтроллеров 8051, 68HC11 или PIC. В этом случае следует прибегнуть к помощи программного обеспечения, называемого обычно симулятором или программным эмулятором, которое воспроизводит поведение микроконтроллера, используя компьютер с процессором любого типа.

Кроме этого, чисто программного подхода, существует и аппаратное решение проблемы – эмулятор.

Эмулятор – устройство, которое подключается к компьютеру, используемому для ассемблирования, и имеет специальный кабель с разъемом, который можно вставить вместо СППЗУ или микроконтроллера в панельку на печатной плате разрабатываемого устройства. При этом исполняемый код содержится во встроенной оперативной памяти эмулятора. Тестируемая система считывает программу именно оттуда, словно это обычное СППЗУ, а с помощью ПО системы разработки легко осуществить все необходимые изменения в исполняемом коде.

Таким образом можно установить неразрывную взаимосвязь между исходным и исполняемым кодами, что значительно ускоряет процесс разработки и отладки и уменьшает количество циклов стирания.

Специальные эмуляторы существуют для большинства микропроцессоров и микроконтроллеров, но есть и более универсальные модели, способные работать с различными семействами и типами этих радиоэлементов.

Если вы не хотите использовать ассемблер, можно работать с языками более высокого уровня, такими как С или Pascal. При подобном выборе необходимо программное обеспечение, называемое компилятором (или кросс-компилятором), которое переведет исходный текст программы на ассемблер, после чего ее легко преобразовать в исполняемый код обычным способом.

Построенный в таком случае код не всегда столь же эффективен, как код, полученный при программировании на языке ассемблера. Часто он оказывается более громоздким и обрабатывается медленнее, однако сейчас значительно увеличено быстродействие программ на языках высокого уровня (в частности, на С).

Ниже приведен пример короткой программы, написанной на языке Pascal для микроконтроллеров семейства 8051. (Те, кто знаком с популярным языком Turbo Pascal, практически не испытают затруднений в работе с этой версией.) Отметим обращение к функциям, свойственным только микроконтроллерам, в частности обращения к его счетчикам-таймерам.

```
var
  timer,frac: word;

procedure interrupt timer0;
  begin
    inc(frac);
    if frac=4000 then
    begin
      inc(timer);
      frac:=0
    end;
    TF0:=false
  end;
begin
  timer:=0;
  TR0:=true;
  ET0:=true; { разрешение прерываний от timer0 }
  TMOD:=2; { Timer0 M1 true: 8-разрядный таймер с автозагрузкой }
  TH0:=6;  { прерывание каждые 250 мс (если тактовая частота 12 МГц) }
  frac:=0;
  EA:=true;  { разрешить прерывания }
  repeat
    (* таймер может что-нибудь делать. *)
    p1.0:=((timer and 1)=0);
  until false;
end.
```

Далее приведено начало листинга перетрансляции на ассемблер, который выполнен компилятором Pascal. Исходные выражения на языке Pascal представлены в виде комментариев, а соответствующие стандартные команды ассемблера – обычными мнемоническими обозначениями (так же поступил бы программист, если бы готовил эту программу непосредственно на ассемблере).

```
    , MicroControllerPascal compiler, vers. 3.09 copyright (C) KSC 1986, 1990
    , With code generation for INTEL 8051
$DEBUG

      NAME          DEMO
      USING     3
      USING     2
      USING     1
      USING     0

  ; timer,frac: word;
?DEMO?D SEGMENT DATA
  RSEG          ?DEMO?D
TIMER:          DS  02H
  RSEG          ?DEMO?D
FRAC: DS        02H
  ;
  ,procedure interrupt timer0,
  ; begin

?DEMO?C         SEGMENT CODE
                    RSEG   ?DEMO?C

TOINT:
      PUSH      PSW
      PUSH      ACC
      PUSH      B
      PUSH      DPL
      PUSH      DPH
      MOV       PSV,#010H
 ;inc(frac);
      INC       FRAC
      MOV       A,#00H
      CJNE      A,FRAC,$+5
      INC       FRAC+01H
 ;if frac=4000 then
      MOV       A,FRAC+01H
      CJNE      A,#15.$+0BH
      MOV       A,FRAC
      CJNE      A,#160,$+06H
      LJMP      ?L0
      LJMP      ?L1
?L0:
 ;begin
 ;inc(timer);
      INC       TIMER
      MOV       A.#00H
      CJNE      A,TIMER,$+5
```

```
      INC         TIMER+01H
 ;frac:=0
 ;end,
      MOV         FRAC,#LOW(00H)
      MOV         FRAC+01H.#HIGH(00H)
?L1:
 ;TF0 =false
 ;end;
      CLR         TF0
      POP         DPH
```

Следовательно, эта программа может быть обработана, например, ассемблером-линкером, предназначенным для семейства 8051, причем так, чтобы в один или два этапа получить исполняемый код, пример которого представлен ниже (в виде файла в формате Intel hex).

```
.0200000080245A
:03000B0002002CC4
:200026001200990200 6EC0D0C0E0C0F0C082C08375D01005237400B523020524E524B40F7A
:2000460008E523B4A003020052020061052174 00BB210205227523007524 00C28DD083D040
·2000660082D0F0D0E0D0D0327521007522 00D28CD2A9758902758C0675230075240 0D2AFF7
:20008600E5215401C394006002740 1F413929080EF80FE752526D000D001852581C001C0AE
:0200A600002236
:00000001FF
```

Указанная последовательность байтов может быть непосредственно «зашита» в СППЗУ микроконтроллера. В таком случае говорят о коде «прошивки» – в отличие от кода, генерируемого обычными, менее специализированными компиляторами, который может быть выполнен только под управлением операционной системы компьютера (например, MS DOS).

Следующий пример показывает, как машинный код при помощи программы-дизассемблера может быть представлен в более ясной форме – в стандартных мнемониках. Сразу отметим, что синтаксис команд не соответствует приведенному в исходном тексте программы, но это правильно.

Дизассемблирование файла в формате Intel hex:

```
AMS 8032 PROGRAM lister  DATE 1/2/1980 TIME 1:37 page 1
COPYRIGHT 1989       A M.S.

PC    MACHN   OPCD    OPERAND
0026  120099  LCALL   0099
0029  02006E  LJMP    006E
002C  C0D0    PUSH    D0
002E  C0E0    PUSH    E0
```

```
0030 C0F0   PUSH  F0
0032 C082   PUSH  82
0034 C083   PUSH  83
0036 75D010 MOV   D0,#10
0039 0523   INC   23
003B 7400   MOV   A, #00
003D B52302 CJNE  A,23,0042
0040 0524   INC   24
0042 E524   MOV   A.24
0044 B40F08 CJNE  A,#0F,004F
0047 E523   MOV   A,23
0049 B4A003 CJNE  A,#A0,004F
004C 020052 LJMP  0052
004F 020061 LJMP  0061
0052 0521   INC   21
0054 7400   MOV   A,#00
0056 B52102 CJNE  A.21.005B
0059 0522   INC   22
005B 752300 MOV   23,#00
005E 752400 MOV   24,#00
0061 C28D   CLR   8D
0063 D083   POP   83
0065 D082   POP   82
0067 D0F0   POP   F0
0069 D0E0   POP   E0
006B D0D0   POP   D0
006D 32     RETI
006E 752100 MOV   21,#00
0071 752200 MOV   22,#00
0074 D28C   SETB  8C
0076 D2A9   SETB  A9
0078 758902 MOV   89,#02
```

В полученном листинге используются произвольные имена переменных и меток, значения которых вместо программиста определяет программа-ассемблер (или компилятор). Например, компилятор «решил», что PSW соответствует регистру D0, а DPL – адресу памяти 82h; о таком выборе дизассемблер, естественно, не имеет никакой возможности узнать.

Вероятно, эти примеры, намеренно извлеченные из очень простой программы, покажутся большинству читателей абсолютно недоступными и бессмысленными. В этом нет ничего удивительного, потому что браться за программирование для микроконтроллеров можно, только предварительно изучив тот или иной язык программирования (ассемблер, Pascal или С) и обладая глубокими знаниями в области аппаратного обеспечения микроконтроллера.

Правильно программировать микроконтроллер намного сложнее, чем разрабатывать GAL или «прошивать» в СППЗУ программу логического автомата.

Назначение этой книги – представить весь арсенал программируемых радиоэлементов, не особенно вдаваясь в подробности механизмов программирования какого-либо микроконтроллера.

ПРОГРАММАТОРЫ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Итогом рассмотренного нами процесса разработки является загрузка программы в СППЗУ микроконтроллера.

Дальнейшие действия будут разными для разных радиоэлементов: прогон через промежуточное СППЗУ и использование специальной программы-загрузчика для микроконтроллера 68705P3, применение специального программатора для микроконтроллера PIC 16C54, подключение простого адаптера к программатору СППЗУ для микроконтроллеров 8751 и 8753 или применение загрузки «на плате» для микроконтроллера PIC16C84.

«Универсальные» промышленные программаторы обычно могут поддерживать большое количество микроконтроллеров, но не все, хотя цена на такие программаторы достаточно высокая.

МИКРОКОНТРОЛЛЕРЫ PIC

Термин «микроконтроллер» обычно ассоциируется с микросхемой, имеющей 40 или 28 выводов, а также многочисленные порты ввода/вывода. Предполагается, что подобная микросхема не всегда достаточно быстродействующая, требующая к тому же дорогостоящих систем разработки – словом, далеко не идеал для ряда широко распространенных приложений, в которых необходимо объединить малые габариты, требуемые параметры и низкую стоимость.

Наиболее популярны на сегодняшний день микроконтроллеры семейства PIC16Cxx компании Microchip. Простота и доступность этого радиоэлемента, способного конкурировать с более совершенными изделиями, вызвана тем, что подобные микроконтроллеры снабжены ограниченным количеством линий ввода/вывода и размещаются в пластмассовом корпусе ОП с малым числом выводов. Они способны обходиться без кварцевого резонатора, вместо которого используются керамические резонаторы или RC-цепи, и легко программируются. Такие радиоэлементы можно найти в мобильных телефонах, радиоприемниках и магнитолах, системах контроля доступа и кодированного телеуправления, различных датчиках. Они

встречаются везде, начиная от автомобилей и электробытовых приборов и заканчивая «конструкторами» – наборами деталей, в которых они могут легко заменить обычные микросхемы. (Если стереть маркировку микроконтроллера, будет обеспечена и дополнительная защита от незаконного повторения.)

Основные области их применения – это все приложения, которые будут слишком сложными, дорогими или громоздкими, если разрабатывать их с использованием обычных цифровых схем или микроконтроллера. Таким образом, переход к микроконтроллеру ОП позволяет значительно усовершенствовать новые приложения и улучшить технические параметры целого ряда уже существующих товаров.

Микроконтроллеры семейства PIC16Cxx сочетают в себе свойства традиционной цифровой техники, программируемых схем и микропроцессоров. Ими пользуются всякий раз, когда с их помощью можно выпустить более высокопроизводительный и компактный продукт, причем быстрее и за меньшую цену.

Семейство PIC16Cxx основано на двух базовых аппаратных конфигурациях с 12 или с 20 линиями ввода/вывода соответственно.

PIC16C54 – наиболее простой прибор, размещенный в корпусе DIP или SOIC с 18 выводами. Он имеет 12 линий ввода/вывода, ОЗУ емкостью 32 байта и СППЗУ емкостью 512×12 бит (что уже является оригинальным решением).

Совместимый с ним на уровне выводов микроконтроллер PIC16C56 оснащен СППЗУ с удвоенным объемом памяти.

Размещенный в корпусе DIP или SOIC с 28 выводами, микроконтроллер PIC16C55 располагает 20 линиями I/O, 32-байтным ОЗУ и СППЗУ емкостью 512×12 бит.

Его вариант PIC16C57 имеет СППЗУ с увеличенным в четыре раза объемом памяти.

PIC16C71 – более новый и заметно улучшенный микроконтроллер. В частности, здесь добавлены аналоговые входы (встроен АЦП) и вместо 12-разрядных используются 14-разрядные команды, которые записываются в 1024 ячейки ПЗУ.

Появившийся недавно, но уже очень распространенный PIC16C84 изготовлен по технологии ЭСППЗУ. Несмотря на то что он размещен в пластмассовом корпусе, его можно моментально «очистить» и тотчас же перепрограммировать. Часть его энергонезависимой памяти доступна для данных, которые не должны теряться при отключении питания.

Наиболее распространенное исполнение версий с СППЗУ – пластмассовый корпус (ОП – OTP), но существуют микросхемы и в керамическом корпусе с кварцевым окошком (УФ СППЗУ – UVPROM),

используемые в процессе разработки и отладки. Есть и возможность программирования на производстве (quick turn production, QTP – быстрый запуск в производство), что гораздо выгоднее изготовления масочных ПЗУ. Переход от прототипа к серии происходит быстрее и без значительных расходов, а возможные изменения программы вызовут минимальные затруднения.

Все эти радиоэлементы выполнены по технологии КМОП и питаются от напряжения 5 или 3 В (версии с пониженным потреблением энергии), ток потребления от 15 мкА до 2 мА в зависимости от тактовой частоты. Разрешается включить дежурный режим (standby), причем ток составит менее 3 мкА. Кроме того, они выпускаются с различными вариантами тактовых генераторов (0–20 МГц).

Эти радиоэлементы приспособлены для работы в системах с автономным питанием, в портативных или даже карманных устройствах. Все микроконтроллеры PIC располагают встроенным счетчиком-таймером, вход синхронизации которого подключен к специально предназначенному выводу, но может работать и от общего тактового генератора. Наконец, допустимо использовать программируемый таймер аварийного сброса – watch dog.

Основная особенность PIC – их RISC-архитектура (Reduced Instruction Set Computer – ограниченная система команд компьютера), которая по отношению к классической конфигурации CISC имеет следующие преимущества:

- более компактный код (вдвое);
- возросшее быстродействие (вчетверо);
- ускорение процесса разработки (приблизительно 30%).

Вместо 50–100 команд по несколько байтов и нескольких рабочих циклов обычных микроконтроллеров язык PIC16C5x ограничивается 33 командами, исполняемыми за один рабочий цикл (за исключением команд условных переходов, использующих два периода) и занимающими в памяти программ (12 разрядов вместо 8) только одно слово. Микроконтроллер PIC16C71 имеет 35 команд.

Преимущество рассматриваемых микроконтроллеров особенно заметно, поскольку многие программисты практически не используют наиболее сложные команды CISC-процессоров из-за того, что они слишком различаются у разных процессоров.

Конечно, применяется и программное обеспечение, написанное на языке высокого уровня (C), а затем откомпилированное, но в этом случае код получается более громоздким и выполняется медленнее, чем необходимо.

С набором команд, ограниченным только самыми главными операциями, становится довольно просто писать непосредственно на ассемблере компактные и эффективные программы, причем делать это быстро. Этот принцип поддерживается особенной архитектурой, называемой Гарвардской, в которой две разные шины выделены для данных и для команд, что полностью устраняет узкое место в архитектурах CISC-машин.

Параллельно с этим используется система, именуемая Pipe-Line (конвейер), что позволяет выполнять команду одновременно с подготовкой следующей, извлекая максимум из каждого тактового периода. Рабочие показатели достигают 5 MIPS (миллионы команд в секунду), что делает возможным оригинальный подход к определенным задачам, которые выгодно обрабатывать в реальном масштабе времени.

При использовании микроконтроллеров PIC допустимо, например, заменять громоздкие таблицы данных алоритмами быстрых вычислений практически так же, как это делалось бы в устройствах на DSP (Digital Signal Processor – цифровой сигнальный процессор).

Микроконтроллеры PIC вполне оправдали существование специфических средств разработки, которые можно приобрести по весьма низким ценам.

Коммерческий успех дешевых и удобных в программировании микроконтроллеров привел к тому, что компания Motorola, в частности, решила бесплатно предоставить своим клиентам необходимое программное обеспечение для сборки собственных простых программаторов; компания SGS-Thomson предпочла выпустить в продажу недорогой полный «комплект разработчика».

Фирма Microchip проводит политику, которая удачно объединяет эти два подхода: комплект Starter Kit, который называется PICSTART-16B, можно приобрести по весьма разумной цене, особенно учитывая его содержимое: специальный программатор, подключаемый к ПК (см. рис. 4.3), программное обеспечение (ассемблер, симулятор и программатор), а также полный комплект документации и несколько корпусов микроконтроллеров PIC, стираемых ультрафиолетовыми лучами.

Разнообразное и практически эквивалентное программное обеспечение можно загрузить и бесплатно (с BBS или из Internet), к тому же алгоритмы программирования были в конце концов опубликованы. В главе 5 мы покажем, до какой степени простой может оказаться, например, сборка программатора для микроконтроллера PIC16C84.

ПО, управляющее программатором Picstart, удобно и просто в использовании благодаря «оконной» структуре, напоминающей Windows, достаточно быстрой и не занимающей много места в памяти.

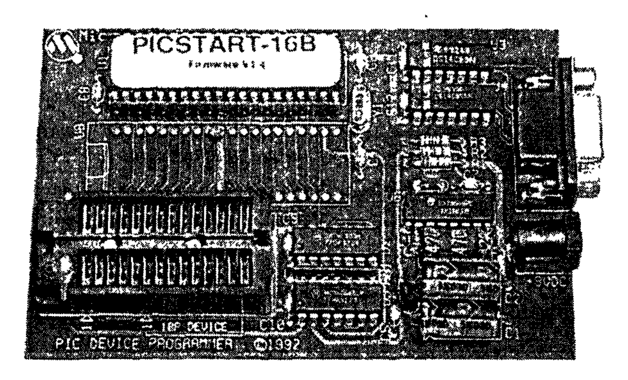


Рис 4 3. Внешний вид программатора из набора PICSTART-16B

Оно позволяет загружать полученный после работы программы-ассемблера шестнадцатеричный файл, редактировать его в случае необходимости, а затем «зашить» в PIC. Понятно, что предварительно надо подготовить так называемые биты конфигурации: бит таймера автозагрузки (watch dog), защиты от копирования, типа тактового генератора, идентификатора. После завершения программирования выполняется контрольное считывание, но можно считывать или проверять содержимое памяти PIC в любое время и контролировать чистоту памяти нового или стертого микроконтроллера.

Что же касается программы-ассемблера и симулятора, то они на первый взгляд ничем особенным не отличаются. Функционируя в режиме командной строки, они напоминают о том времени, когда вся работа происходила на телетайпах. Однако при более внимательном рассмотрении становится понятно, что это мощные средства, не уступающие продуктам, цена которых в пять-десять раз превышает цену всего этого комплекта. Поэтому можно порекомендовать пользователю хорошо изучить определенный набор сокращенных команд (если только он не предпочитает постоянно пользоваться системой справочной информации) вместо того, чтобы щелкать кнопками мыши. Вероятно, именно этим и отличаются настоящие программисты-профессионалы от пользователей.

Симулятор позволяет выполнить искусственный прогон программы, предварительно обработанной ассемблером, в несколько десятков или сотен раз медленнее, чем такое происходило бы в реальности. В сочетании с возможностью расставлять точки останова и выводить на экран содержимое различных регистров микроконтроллера это позволяет создавать программы для работы в режиме реального времени с точностью почти до одного тактового периода.

ГЛАВА СТРАНИЦА

5 ИЗГОТОВЛЕНИЕ ПРОГРАММАТОРОВ

Теперь необходимо перейти к практическим действиям: начать программировать радиоэлементы. Но для этого необходимо соответствующее оборудование – один или несколько программаторов. Наиболее эффективным и гибким будет использование программатора, подключаемого к обычному ПК, и нескольких мощных программ, разработанных для этих целей.

ПРОГРАММАТОР СППЗУ

На сегодняшний день совместимая с IBM PC микро-ЭВМ является очень распространенным и недорогим оборудованием, используемым в самых разных областях.

Для записи и считывания содержимого СППЗУ и других программируемых радиоэлементов вполне достаточно простого и дешевого устройства, при этом нет никакой необходимости разбираться в конфигурации компьютера: это осуществляется при помощи программ, часто занимающих несколько десятков строк.

По существу, программирование СППЗУ (или радиоэлемента, основанного на технологии СППЗУ) является простой операцией: достаточно по очереди перебрать все адреса, в которые предполагается произвести запись, и после выставления байта данных для программирования подать на специальный вывод правильно откалиброванный импульс. Конечно, соблюдая при этом величины напряжений питания, указанные производителем.

Данные, которые подлежат записи, могут быть уже готовыми и располагаться в другом СППЗУ (причем необязательно того же типа), на дискете, на бумаге и даже в голове оператора.

Сложность заключается лишь в том; что СППЗУ разных типов отличаются друг от друга цоколевками, напряжениями питания и циклограммами программирования, что приводит к конфликтам, не всегда разрешимым.

Программатор, автоматически учитывающий все возможные случаи, обязательно будет сложным аппаратом с адаптивной программной перенастройкой.

Для того чтобы он был быстродействующим, необходимо работать непосредственно с шинами данных и адресом главного компьютера при помощи схем PIA.

Специалист по электронике способен безошибочно подключить несколько кабелей и подождать несколько минут, пока программирование завершится. Поэтому при подключении программатора можно

использовать параллельный порт для принтера, который есть практически на каждом ПК.

Его восемь линий данных позволяют передавать непосредственно байты, предназначенные для записи, в то время как на линии STROBE можно формировать импульсы программирования, длительность которых будет определяться матобеспечением (прецизионные таймеры и моновибраторы, таким образом, совершенно не нужны). Есть несколько входных линий (по крайней мере, линии ACK и BUSY). В дальнейшем их можно использовать для считывания информации из СППЗУ с целью проверки правильности записи, дуплицирования или даже модифицирования содержимого.

Но возможность организовать адресную шину при помощи параллельного порта отсутствует, поскольку такой порт слишком «узок». В большинстве случаев СППЗУ надо программировать, начиная с его первого адреса и до последнего или же частично, но с последовательным порядком адресов. Простой счетчик, использующий импульсы программирования как тактовый сигнал, способен легко перебрать все вероятные состояния адресной шины при условии, что он будет сброшен в нуль синхронно с запуском программного обеспечения. А это сделать совсем не трудно.

Что касается питания, то в любой более или менее оборудованной лаборатории найдется все необходимое для получения напряжения 5 В и второго напряжения в диапазоне 12–25 В, причем оба источника рассчитаны на ток не больший, чем несколько десятков миллиампер. Можно в случае необходимости собрать специальное питающее устройство, но это совсем не обязательно.

На рис. 5.1 представлены КМОП счетчик типа 4040 и сшестеренный инвертор типа 4049 – два действительно необходимых для работы активных радиоэлемента.

Восемь резисторов сопротивлением 470 Ом (или одна резисторная сборка в исполнении DIL) подключены к шине данных, чтобы избежать проблем в случае конфликта: они ограничивают ток, который может протекать, если запоминающее устройство случайно окажется в режиме чтения.

Восемь линий данных, защищенные подобным образом, подключаются непосредственно к панельке, так как разводка их одинакова для всех запоминающих устройств, соответствующих стандарту Bytewide.

В распоряжение пользователя одновременно предоставляются парафазные сигналы STROBE, чтобы на запоминающее устройство в зависимости от его типа мог подаваться импульс соответствующей

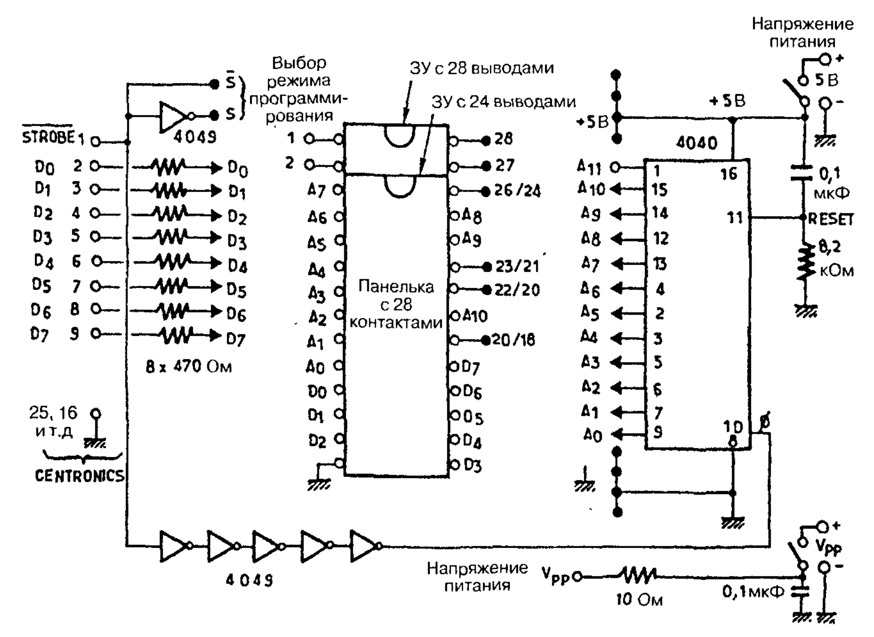


Рис. 5.1. Принципиальная схема программатора

полярности. Этот импульс инвертируется и задерживается каскадом из пяти инверторов, после чего поступает на тактовый вход счетчика 4040, начальная установка которого при включении питания обеспечивается RC-цепочкой.

Двенадцать адресных линий соответствуют выходам счетчика 4040, и этого достаточно для запоминающих устройств емкостью до 4 Кб (2^{12} = 4096), то есть для СППЗУ типа 2732. Линии А0-А10 являются общими для всех СППЗУ, начиная от 2716 и до 27256 и более новых; они подключаются непосредственно к панельке. С помощью перемычки линия А11 может подключаться к выводам 23/21 панельки, начиная с микросхем типа 2732 и далее.

Чтобы использовать линии А12–А14, надо скоммутировать надлежащим образом выводы 1, 2, 27 и 28 панельки, которые оставались незанятыми для микросхем 2716 и 2732.

На рис. 5.2 отображены те соединения, которые необходимо выполнить при программировании наиболее распространенных типов СППЗУ; здесь же показано, как относительно простые манипуляции с проволочными перемычками позволяют программировать

микросхемы 2764 двумя блоками по 4 Кб, микросхемы 27128 – в четыре этапа, а микросхемы 27256 – в восемь этапов.

Эта процедура будет слишком сложной при работе с запоминающими устройствами полезным объемом свыше 8 Кб, но иметь такую возможность желательно (поэтому лучше использовать панельку с 28 выводами вместо 24). Может потребоваться работа и с 27C512 или 27C010, где объем записанных данных не превышает объема микросхемы 2716.

И наконец, два выключателя позволяют управлять напряжениями V_{CC} и V_{PP} по соответствующему запросу, формируемому программой.

Известно, что появление (даже на доли секунды!) напряжения V_{PP} при выключенном V_{CC} выводит из строя практически любое запоминающее устройство. Поэтому мы настоятельно рекомендуем читателям соблюдать соответствующий порядок. А столь простая схема без труда размещается на печатной плате, топология которой показана на рис. 5.3.

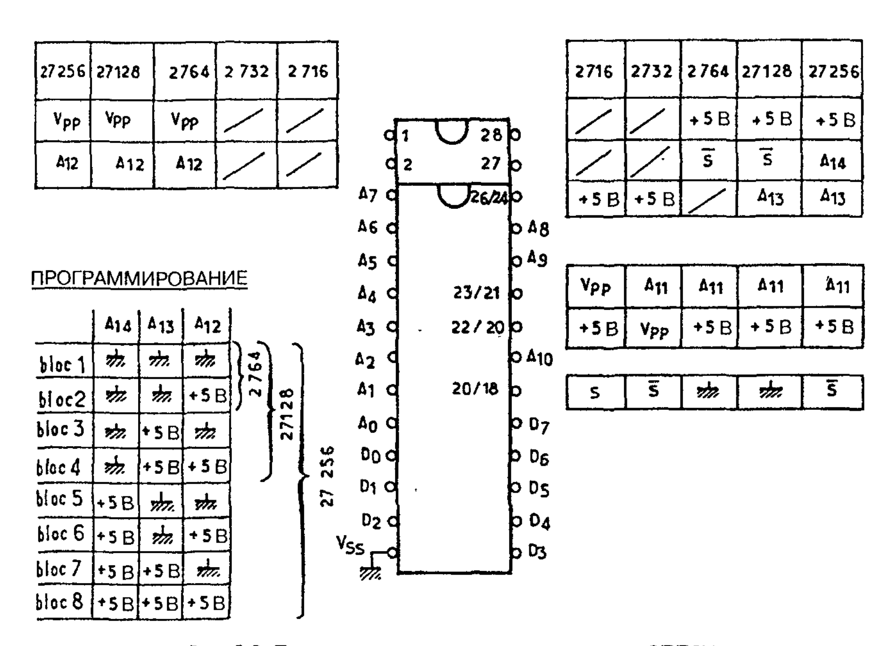


Рис. 5.2. Порядок подключения основных типов СППЗУ

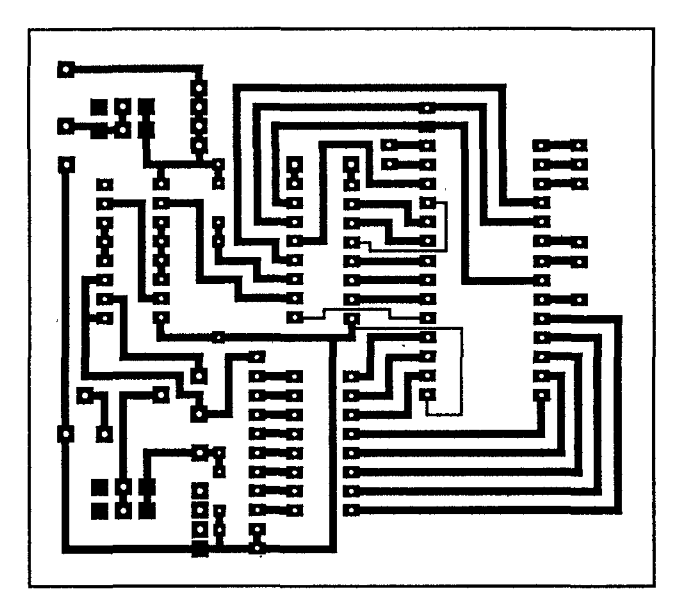


Рис. 5.3. Топология печатной платы программатора

Оригинальность конструкции заключается в том, что сторона элементов (см. рис. 5.4) используется достаточно слабо: на ней находятся две интегральные микросхемы, резисторы и конденсаторы, одна перемычка, а также контактные площадки для подключения проводов и кабелей (источников питания V_{CC} и V_{PP} и плоского десятижильного кабеля, на одной стороне которого распаяна стандартная вилка Centronics, идентичная разъему принтера).

Именно со стороны печатных проводников (см. рис. 5.6) припаивают панельку для микросхем с 28 контактами (гиперболическими или цанговыми), причем лучше всего – ZIF-панельку (с нулевым усилием установки), два выключателя и, самое главное, 20 гнезд, также с гиперболическими или цанговыми контактами, сделанных из «разрезной колодки» или изготовленных из старой панельки для интегральных микросхем. На рис. 5.5 показан внешний вид программатора со стороны элементов, а на рис. 5.7 – со стороны печати.

Именно эти гиперболические или цанговые контакты будут служить гнездами для конфигурирования панельки: вместо проводов с обычными 4-миллиметровыми штекерами используют куски жесткого изолированного монтажного провода диаметром 0,6 мм, очищенные от изоляции с каждого конца на длину 3–4 мм. Это дешево и практично.

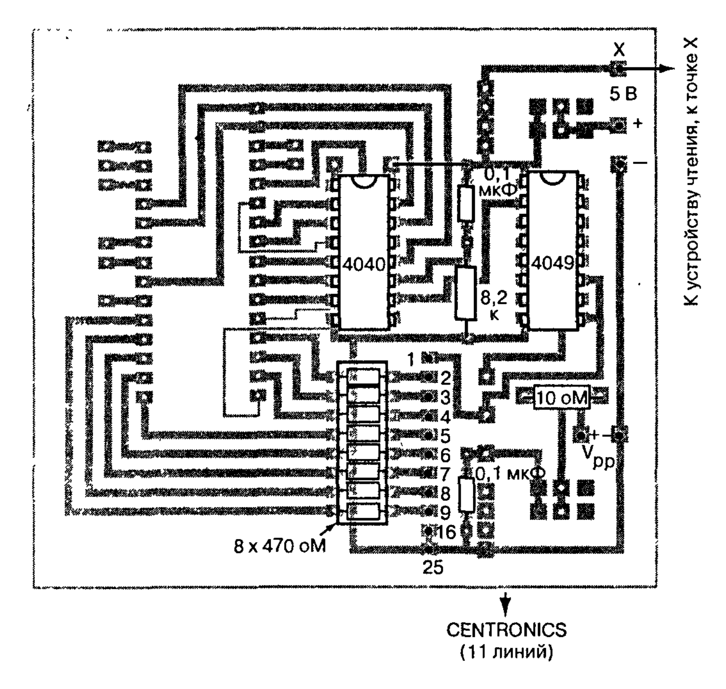


Рис. 5.4. Схема размещения элементов программатора

Таблица к рис. 5.4

Перечень элементов		
Наименование	**Тип/Номинал**	**Примечание**
Резисторы	10 Ом	
	8,2 кОм	
	470 Ом	8 шт. или 1 схема DIL
Конденсаторы	0,1 мкФ	
Интегральные микросхемы	CD 4040	
	CD 4049	
Прочее	Панелька с 28 гиперболическими контактами	
	Выключатель однополюсный (2 шт.)	
	Разрезная колодка с гиперболическими контактами	
	Разъем Centronics (модель для принтера)	
	Кабель из 11 проводников	

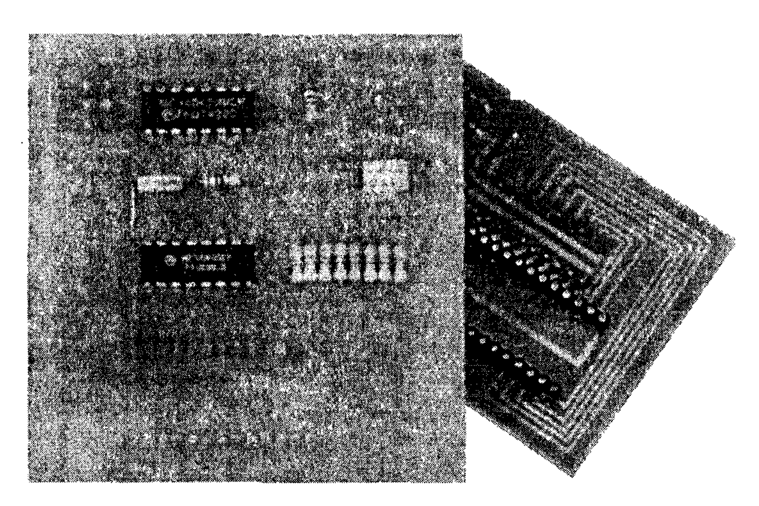


Рис. 5 5. Внешний вид программатора со стороны элементов

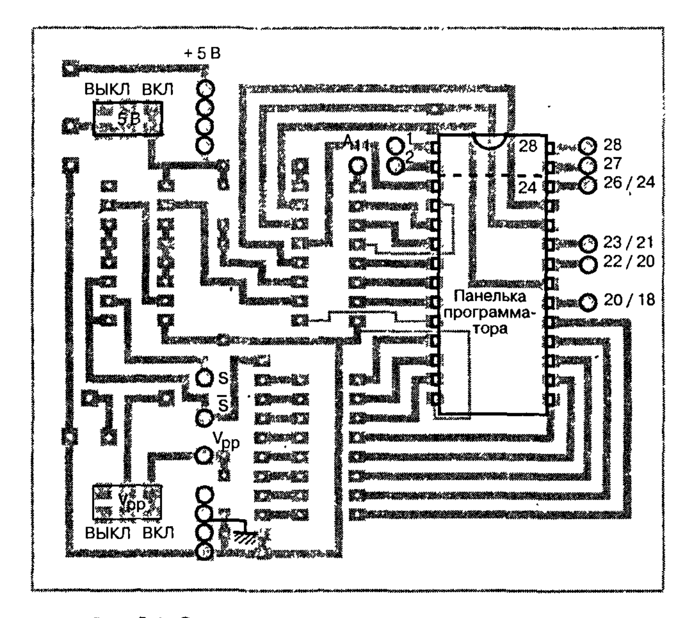


Рис. 5.6. Схема размещения элементов со стороны печати

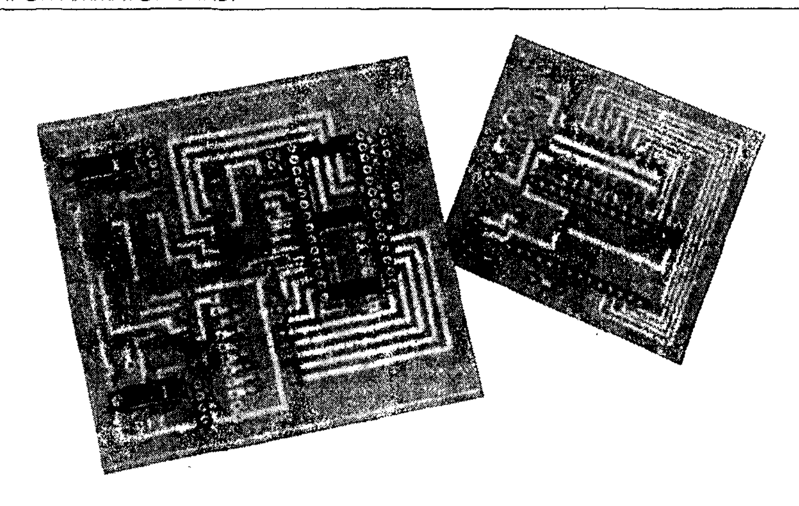


Рис. 5.7. Внешний вид программатора со стороны печати

Дополнительное преимущество такой конструкции с двусторонним монтажом заключается в том, что плата может быть установлена непосредственно на лицевой панели подходящего корпуса с помощью обычных гаек в качестве монтажных стоек. Панелька, контакты и выключатели практически одинаковы по высоте, что делает конструкцию удобной и надежной одновременно.

Программное обеспечение

Предлагаемая концепция максимального упрощения аппаратного обеспечения подразумевает перенос всех основных задач на ПО.

Программы, в частности, должны непосредственно управлять портами ввода-вывода, работающими с разъемом принтера, и, следовательно, поддерживать связь с ними через определенные адреса. Все программы, приведенные здесь, предназначены для работы с портом LPT1 и поэтому должны быть изменены, если используется другой порт.

Для каждой программы предлагаются две версии: одна написана на языке Basic, а другая – на Turbo Pascal. Этот популярный язык значительно быстрее Basic, но не входит в стандартный набор программного обеспечения.

Программа PROGROM (progrom.bas и progrom.pas соответственно) будет искать данные для программирования в текстовом файле, представленном на дискете (с расширением .rom по умолчанию), в котором каждый байт представлен в виде явного десятичного значения (см. главу 2) и имеет название free format decimal).

```
10 REM -- PROGROM.BAS --
20 CLS:PRINT"ОТКЛЮЧИТЕ ПИТАНИЕ"
30 PRINT"имя файла для прошивки СППЗУ?"
40 INPUT F$:F$=F$+".ROM"
50 OPEN"i",#1, F$
60 DIM M(4096)
70 CLS:PRINT:PRINT"-- ОСУЩЕСТВЛЯЕТСЯ ЧТЕНИЕ ФАЙЛА --"
80 F=0
90 IF EOF(1) THEN 130
100 INPUT#1,M(F)
110 F=F+1
120 GOTO 90
130 CLOSE#1:CLS
140 PRINT"ДЛИТЕЛЬНОСТЬ ИМПУЛЬСА в миллисекундах?"
150 INPUT M
160 IF M>50 THEN M=50
170 IF M<5 THEN M=5
180 CLS:PRINT"подключите чистое СППЗУ"
190 PRINT"минимум на";F;" байтов"
200 PRINT"затем нажмите ENTER"
210 INPUT Z$:CLS
220 PRINT"подайте напряжение +5 В, затем нажмите ENTER"
230 INPUT Z$
240 PRINT"подайте напряжение Vpp, затем нажмите ENTER "
250 PRINT:PRINT"ПРОВЕРЬТЕ ВЕЛИЧИНУ Vpp!"
260 INPUT Z$
270 CLS:PRINT:PRINT"--ОСУЩЕСТВЛЯЕТСЯ ПРОГРАММИРОВАНИЕ--"
280 FOR G=0 TO F-1
290 D=M(G)
300 OUT 888,D
310 OUT 890,1
320 FOR T=1 TO M: NEXT T
330 OUT 888,255
340 OUT 890,0
350 NEXT G
360 PRINT:PRINT:PRINT:BEEP:CLS
370 PRINT"ОТКЛЮЧИТЕ Vpp, затем нажмите ENTER"
380 INPUT Z$: PRINT
390 OUT 888,0
400 PRINT"ОТКЛЮЧИТЕ +5 В, затем нажмите ENTER"
410 INPUT Z$
420 CLS:PRINT"ВЫНЬТЕ СППЗУ"
430 PRINT:PRINT"ВОЗОБНОВИТЬ? Y/N + ENTER"
440 INPUT 2$
450 IF Z$="Y" OR ZS="y" THEN 180
460 END
470 REM (c)1991 Patrick GUEULLE
```

Программа на языке Turbo Pascal

```
        program PROGROM;
        uses crt;
 var m:array[0..4095] of integer,
     h:string[12];
     f:integer;
     e:boolean;
     g:integer;
     a:text;
     t:integer;
     n:string[1];
procedure Init;
begin
clrscr;
port[888]:=0;
writeln("ВНИМАНИЕ! ОТКЛЮЧИТЕ ПИТАНИЕ");
writeln;
end;
procedure Lecture;
begin
writeln ("ИМЯ ФАЙЛА ДЛЯ ЗАПИСИ В СППЗУ?"),
readln(h);
h:=h+'.rom';
assign(a,h);
reset(a);
clrscr;
writeln("---ОСУЩЕСТВЛЯЕТСЯ ЧТЕНИЕ ФАЙЛА---");
f:=0;
repeat
read(a,m[f]);
e:=seekeof(a);
f:=f+1;
until e=true;
close (a);
end;
procedure Eprom;
begin
clrscr;
writeln("ДЛИТЕЛЬНОСТЬ ИМПУЛЬСА в миллисекундах?");
readln(t);
if t>50 then t:=50;
if t<5 then t:=5;
clrscr;
writeln("ПОДКЛЮЧИТЕ ЧИСТОЕ СППЗУ");
writeln("минимум на",f,"байтов");
writeln("затем нажмите ENTER");
readln;
```

```
clrscr;
writeln('подайте напряжение +5 В, затем нажмите ENTER");
readln;
writeln("подайте Vpp,затем нажмите ENTER");
writeln(" ПРОВЕРЬТЕ ВЕЛИЧИНУ Vpp!");
readln;
clrscr;
writeln("-- ОСУЩЕСТВЛЯЕТСЯ ПРОГРАММИРОВАНИЕ --");
for g:=0 to f-1 do
begin
port[888]:=m[g];
port[890]:=1;
delay(t);
port[888]:=255,
port[890]:=0;
end;
clrscr;
sound(440);
delay(1000);
nosound,
writeln("ОТКЛЮЧИТЕ Vpp, затем нажмите ENTER");
readln;
port[888]:=0;
writeln(" ОТКЛЮЧИТЕ +5 В, затем нажмите ENTER");
readln;
clrscr;
writeln("ВЫНЬТЕ СППЗУ");
writeln;
writeln("ПОВТОРИТЬ? Y/N + ENTER");
readln (n);
if (n='y') or (n='Y') then eprom;
end;
begin
init;
lecture;
eprom;
writeln("КОНЕЦ");
end.
(* COPYRIGHT 1990 Patrick GUEULLE *)
```

Считывая данные, программа строит таблицу из 4096 «ячеек», к которой она сможет обратиться непосредственно во время программирования, не вызывая естественного при любом обращении к диску прерывания, способного повредить СППЗУ несвоевременным увеличением длительности откалиброванного импульса.

Этот файл может быть составлен с помощью любого текстового редактора или получен конвертированием из файла другого формата: двоичного, шестнадцатеричного и т.д.

В состав демонстрационного ПО программатора MQP, которое также содержится на сайте издательства ДМК по адресу http://www.dmk.ru, входит превосходный редактор-транскодер файлов (PDED), полностью приспособленный для такого рода работы. Это условно бесплатное программное обеспечение предполагает выполнение формальной регистрации, если вы захотите регулярно его использовать (для выяснения деталей просмотрите файл read.me). В противном случае программа TRANS.BAS, приведенная ниже, позволяет конвертировать десятичные файлы в двоичные и наоборот.

```
10 REM -- TRANS.BAS --
20 CLS:PRINT"ИМЯ ИСХОДНОГО ФАЙЛА? (с расширением)"
30 INPUT N$
40 PRINT:PRINT"ИМЯ ПОЛУЧЕННОГО ФАЙЛА? (с расширением)"
50`INPUT F$
60 PRINT:PRINT
70 PRINT"Преобразование десятичного файла в двоичный:1"
80 PRINT"Преобразование двоичного файла в десятичный 2"
90 PRINT:PRINT"                    + ENTER"
100 INPUT Z$
110 IF Z$="2" THEN 230
120 IF Z$<>"1" THEN 100
130 OPEN N$ FOR INPUT AS #1
140 OPEN "r",#2,F$,1
150 FIELD#2,1 AS A$
160 CLS:PRINT"Осуществляется преобразование: ";N$; " в ";F$
170 IF EOF(1) THEN 210
180 INPUT#1,M:M$=CHR$(M)
190 RSET A$=M$:PUT#2,F+1
200 F=F+1:GOTO 170
210 CLS:PRINT NS;" преобразованы в";F$
220 END
230 OPEN F$ FOR OUTPUT AS #2
240 OPEN "r",#1,N$,1
250 FIELD#1,1 AS A$
260 CLS:PRINT"Осуществляется преобразование: ";NS;      " в ";F$
270 FOR F=1 TO LOF(1)
280 GET#1,F
2&0 PRINT#2,ASC(A$);
300 NEXT F
310 GOTO 210
320 REM (c)1981 Patrick GUEULLE
```

Если программа на языке Turbo Pascal сама генерирует полностью откалиброванный программирующий импульс, то программа на языке Basic может потребовать «регулировки» в зависимости от тактовой частоты вашего ПК. Поэтому с помощью осциллографа стоит

проконтролировать полученную реальную длительность этого импульса, которая должна равняться 50 мс, и в случае несоответствия этой величине (что возможно на любом ПК выше 286) в программу добавляют строку 175, вносящую корректирующий коэффициент.

СЧИТЫВАЮЩЕЕ УСТРОЙСТВО СППЗУ

Даже если вышеописанный программатор может свободно функционировать один, его объединение со считывающим устройством будет по-настоящему полезно. Во-первых, для переписывания существующих СППЗУ или для перезаписи из микросхем одного типа в другой (например, из четырех 2716 в одну 27С64), во-вторых, в целях проверки правильности теста чистоты новых или стертых СППЗУ и, в-третьих, для контроля после программирования или при ремонте.

Множество программаторов одновременно являются и считывающими устройствами, обычно выполняя две операции без перестановки запоминающего устройства из одной панельки в другую. Концепция, принятая для описываемого программатора (когда конфигурация контактов в панельке определяется с помощью проволочных перемычек и гнезд), исключает такую возможность. Следовательно, нужно собрать отдельно программатор и отдельно считывающее устройство, даже если впоследствии они объединятся в одном корпусе и будут подключены к одному и тому же компьютерному кабелю.

Связь между двумя блоками аппарата будет осуществляться посредством десятичного файла с расширением .rom, записанного на дискету. Это обеспечивает реальную безопасность данных, поскольку их легко проверить, дублировать или редактировать и модернизировать: такой файл действительно может быть обработан с помощью любого редактора текста.

Для того чтобы подключить считывающее устройство к управляющему ПК, мы решили использовать, как и для программатора, параллельный порт принтера, хотя речь теперь идет о вводе данных в компьютер.

Однако в этом случае количество имеющихся линий ввода меньше восьми: $\overline{\text{ACK}}$, BUSY, PE и др. Таким образом, мы практически не имеем выбора: параллельные данные, считываемые из СППЗУ, нужно сделать последовательными, то есть поочередно передавать 8-битные блоки по линиям $\overline{\text{ACK}}$ или BUSY.

На рис. 5.8 показано, как этот метод может быть реализован с помощью совсем небольшого количества радиодеталей: счетчика типа

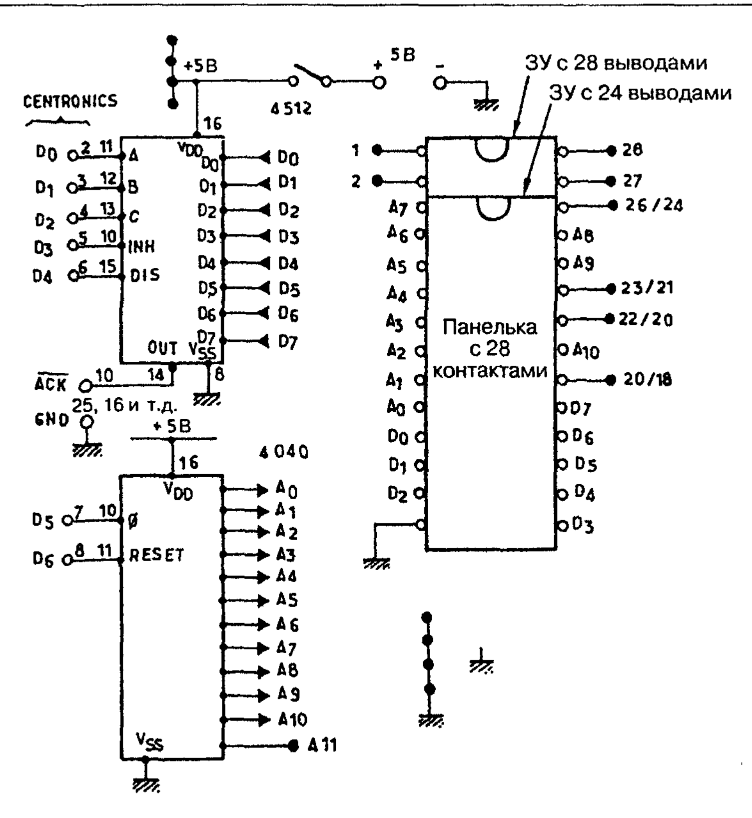


Рис. 5.8. Принципиальная схема считывающего устройства

4040 (чтобы перебирать адреса, как и в программаторе) и селектора данных типа 4512.

Три из восьми линий вывода у «параллельного» разъема служат для того, чтобы указать мультиплексору 4512, какой из восьми разрядов шины данных запоминающего устройства он должен «направить» на линию ввода $\overline{ACK}$ (вывод 10). Две другие из этих линий вывода предназначены для того, чтобы соответственно провести начальную установку счетчика и инкрементировать его, когда надо сменить адрес, в то время как еще две линии обслуживают входы разрешения/стробирования микросхемы 4512. Мы не используем эти входы в данном варианте и, следовательно, удерживаем их программно в состоянии логического нуля, что дает возможность модифицировать схему в будущем.

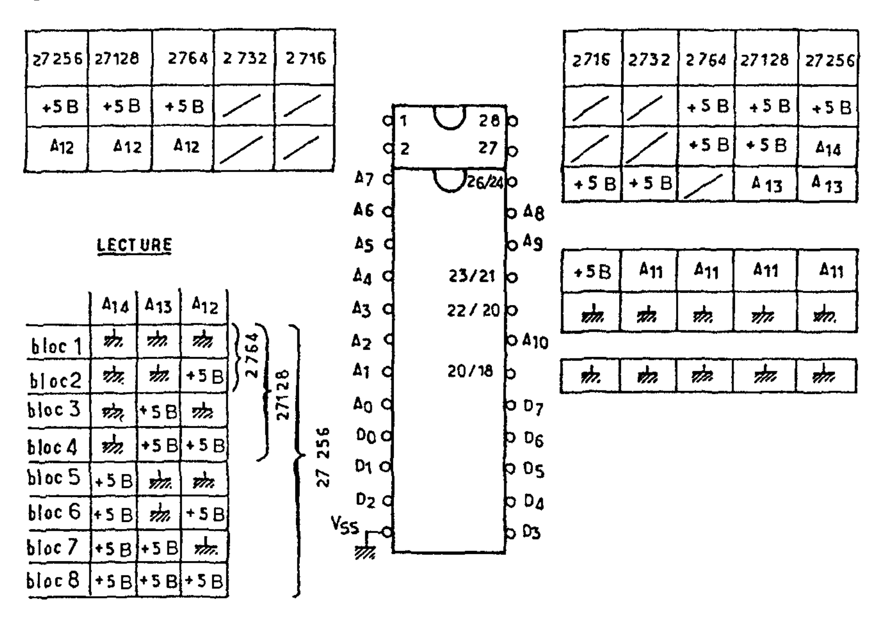


Рис. 5.9. Порядок подключения основных типов СППЗУ

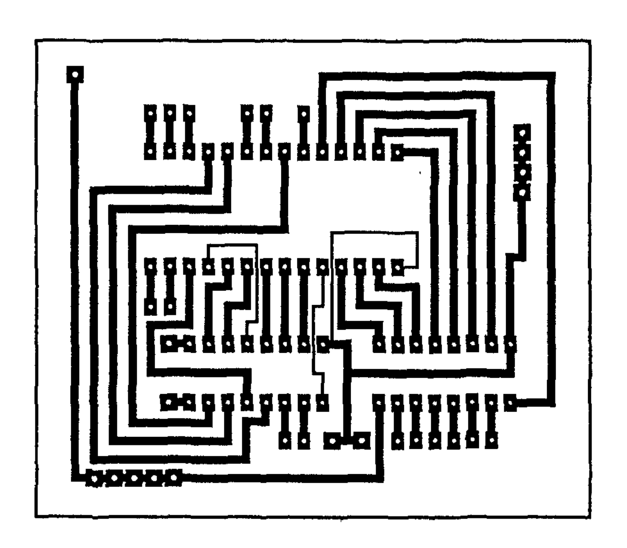


Рис. 5.10. Топология печатной платы считывающего устройства

Во время работы возникает необходимость программно и последовательно считывать эти 8 бит, а затем группировать их в байты, и так для каждого адреса. В программе на языке Basic это происходит достаточно медленно, а на языке Turbo Pascal заметно быстрее.

К панельке СППЗУ подведены линии шин адреса и данных общие для всех распространенных типов микросхем и подключены гиперболические контакты для линий, которые надо выставлять отдельно. Будьте внимательны: на схеме видно, что соединения в режиме чтения

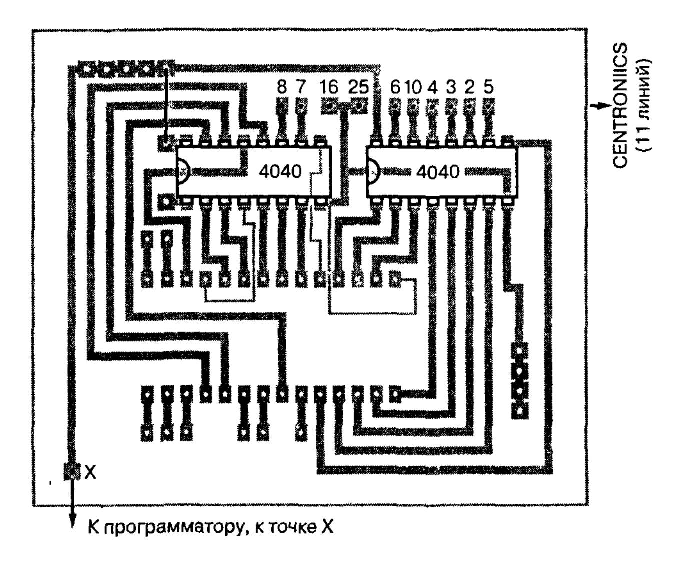


Рис. 5.11. Монтаж со стороны элементов

Таблица к рис. 5.11

Перечень элементов		
Наименование	**Тип/Номинал**	**Примечание**
Интегральные микросхемы	CD 4040	
	CD 4512	
Прочее	Панелька с 28 гиперболическими контактами	
	Разрезная колодка с гиперболическими контактами	
	Шлейф из 11 проводников	

не соответствуют соединениям при программировании, а на линии питания V_{pp} должно быть установлено напряжение 5 В.

На рис. 5.9 приведена вся необходимая информация, для подключения наиболее распространенных типов СППЗУ, но можно сверяться и со спецификациями, и с технической документацией для того, чтобы считывать данные из ПЗУ других типов: масочных ПЗУ, ППЗУ с плавкими перемычками, ЭСППЗУ, флэш и т.д.

Вся схема размещена на маленькой печатной плате, топология которой представлена на рис. 5.10. Ее монтаж выполняется точно также, как и программатора: интегральные микросхемы, перемычка и плоский десятижильный кабель располагаются со стороны элементов, как показано на рис. 5.11, а панелька и изготовленные из «разрезной колодки» гнезда с гиперболическими или цанговыми контактами – со стороны печати (см. рис. 5.12).

Кабель может быть распаян на тот же разъем Centronics, что и кабель программатора, то есть линии данных D0 – D6, линию STROBE и общий провод разрешается включать параллельно. Поэтому не нужно будет производить никаких переключений при смене режимов чтения и программирования: достаточно просто сменить программное обеспечение и переставить СППЗУ из одной панельки в другую, отключив перед этим питание, компьютер в этом случае можно не

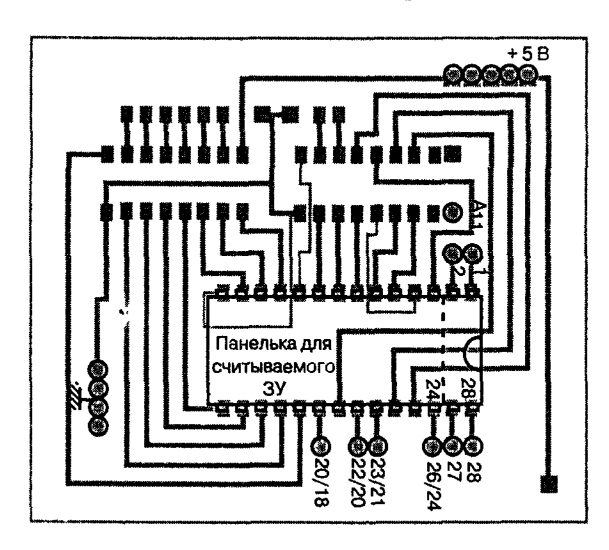


Рис. 5.12. Монтаж со стороны печати

выключать. Не рекомендуется работать с устройством, если микросхемы установлены и в считыватель, и в программатор. Во всяком случае сознательно не предусматривалась прямая передача данных из одной микросхемы в другую минуя файл на компьютере.

Питание на считывающее устройство (+5 В) можно снять с платы программатора уже после тумблера, который таким образом будет использоваться в обоих режимах эксплуатации аппарата, переключать его надо по командам программного обеспечения. С этой целью достаточно соединить точки X двух плат соединительным проводом.

Что касается «корпусирования», то для этого может быть использована любая пластмассовая или металлическая заготовка с площадью лицевой поверхности, достаточной для размещения двух рассмотренных выше плат рядом друг с другом. Корпус с наклонной лицевой панелью – «пультом» – наиболее подходит для этой цели. На лицевой панели, в одном из углов, очень удобно при помощи клейкой ленты с двусторонним липким покрытием приклеить прямоугольный кусок антистатического пенополиуретана или другого проводящего упаковочного материала: на него можно накалывать СППЗУ, подготовленные для размещения в панельках, – элементарная предосторожность, которой не следует пренебрегать.

Программное обеспечение

Управление вышеописанным считывающим устройством требует программного обеспечения, способного управлять последовательной передачей данных, причем именно так, как нам пришлось ее организовать.

Программа LECROM опять представлена в двух вариантах: на языках Basic и Turbo Pascal.

```
10 REM -- LECROM.BAS --
20 CLS
30 PRINT"объем считываемого СППЗУ (в Кб) ?"
40 INPUT K
50 K=(1024*K)-1
60 CLS: PRINT"имя, присвоенное файлу?"
70 INPUT F$
80 OPEN"o",#1,F$+".ROM"
90 CLS:PRINT"ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТКЛЮЧЕНО!"
100 OUT 888,64
110 PRINT"подключите считываемое СППЗУ, затем нажмите ENTER"
120 INPUT Z$:CLS
130 PRINT"подайте +5 В, затем нажмите ENTER"
140 INPUT Z$:CLS
```

```
150 PRINT"--ОСУЩЕСТВЛЯЕТСЯ СЧИТЫВАНИЕ--"
160 FOR G=0 TO K
170 D=0
180 FOR F=0 TO 7
190 OUT 888.F
200 B=INP(889)
210 IF (B AND 64)=64 THEN D=D+(2^F)
220 NEXT F
230 OUT 888,32
240 PRINT#1,D;
250 NEXT G
260 CLS: PRINT"--- ОТКЛЮЧИТЕ +5 В ---"
270 BEEP·END
280 REM (c)1991 Patrick GUEULLE

program LECROM;
uses crt;
var k:integer;
    g:integer;
    f.integer;
    b:integer;
    d:real,
    h:string[12];
    c:boolean;
    a:text;
    x:integer;
    begin
clrscr;
writeln ("объем считываемого СППЗУ (в Кб)?");
readln(k);
k:=(k*1024)-1;
clrscr;
writeln("имя, присвоенное файлу?");
readln(h);
h:=h+'.rom';
assign(a,h);
rewrite(a);
clrscr;
writeln("ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТКЛЮЧЕНО!");
port[888]:=64;
writeln("подключите считываемое СППЗУ, затем нажмите ENTER");
readln;
clrscr;
writeln("подайте +5 В, затем нажмите ENTER");
readln;
clrscr;
writeln("--ОСУЩЕСТВЛЯЕТСЯ СЧИТЫВАНИЕ--");
writeln;
```

```
for g:=0 to k do
begin
d:=0;
for f:=0 to 7 do
begin
port[888]:=f;
b:=port[889];
if (b and 64)=64 then d:=d+(exp(f*ln(2)));
x:=round(d);
end;
port[888]:=32;
write(a,' ",x," " );
end;
sound(440);
delay(1000);
nosound;
clrscr,
writeln("отключите +5 В");
writeln("затем выньте СППЗУ");
close(a);
end.
(* COPYRIGHT 1991 Patrick GUEULLE *)
```

Еще раз повторим: обязательно нужно подождать какое-то время, пока данные из СППЗУ будут считаны, даже если заведомо известно, что для этого достаточно нескольких миллисекунд. К тому же в конце операции предусмотрена подача звукового сигнала.

Программа VEROM (на языках Basic и Turbo Pascal) позволяет проверить соответствие данных в СППЗУ файлу-ссылке, имя которого вы указываете (без расширения .rom, автоматически добавляемого программой). Эта сверка с файлом на дискете дает возможность контролировать операцию программирования, а также устанавливать любое повреждение СППЗУ, если микросхема уже многократно программировалась или подвергалась электрическим ударам.

```
10 REM -- VEROM.BAS --
20 CLS
30 PRINT"имя файла-ссылки?"
40 INPUT F$
50 OPEN"i",#1,F$+".ROM"
60 PRINT"ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТКЛЮЧЕНО"
70 OUT 888.64
80 PRINT"подключите считываемое СППЗУ, затем нажмите ENTER"
90 INPUT Z$
100 PRINT"подайте +5 В, затем нажмите ENTER"
110 INPUT Z$
120 PRINT"---ОСУЩЕСТВЛЯЕТСЯ СРАВНЕНИЕ---"
```

```
130 G=0
140 D=0
150 FOR F=0 TO 7
160 OUT 888,F
170 B=INP(889)
180 IF (B AND 64)=64 THEN D=D+(2^F)
190 NEXT F
200 OUT 888,32
210 INPUT#1,C
220 IF EOF(1) THEN 260
230 IF C<>D THEN PRINT G,D,"вместо", C
240 G=G+1
250 GOTO 140
260 PRINT"-- ОТКЛЮЧИТЕ +5 В --"
270 BEEP:END
280 REM (c)1991 Patrick GUEULLE
```

```
10 REM -- VIROM.BAS --
20 CLS
30 PRINT"обьем СППЗУ(в Кб)?"
40 INPUT K:CLS
50 K=(K*1024)-1
60 PRINT"ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТЛЮЧЕНО"
70 OUT 888.64
80 PRINT"подключите тестируемое СППЗУ, затем нажмите ENTER"
90 INPUT Z$:CLS
100 PRINT"подайте +5 В, затем нажмите ENTER"
110 INPUT Z$
120 PRINT"--ОСУЩЕСТВЛЯЕТСЯ ТЕСТИРОВАНИЕ НА ЧИСТОТУ--"
130 FOR G=0 TO K
140 FOR F=0 TO 7
150 OUT 888,F
160 B=INP(889)
170 IF (B AND 64)=64 THEN 190
180 PRINT"*";:F=7
190 NEXT F
200 OUT 888,32
210 NEXT G
220 PRINT"--- ОТКЛЮЧИТЕ +5 В ---"
230 BEEP:END
240 REM (c)1991 Patrick GUEULLE
```

И наконец, программа VIROM позволяет тестировать чистоту нового или стертого СППЗУ.

```
program VEROM;
uses crt;
var h:string[12];
```

```
    g:integer;
    d:real;
    f:integer;
    b:integer;
    c:integer;
    a:text;
    x:integer;
    e:boolean;
begin
clrscr;
writeln("ИМЯ ФАЙЛА-ССЫЛКИ ?");
readln(h);
h:=h+'.rom';
assign(a,h);
reset(a);
clrscr;
writeln("ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТКЛЮЧЕНО");
port[888]:=64;
writeln("подключите считываемое СППЗУ, затем нажмите ENTER ");
readln;
clrscr;
writeln("подайте +5 В, затем нажмите ENTER");
readln;
clrscr;
writeln("---ОСУЩЕСТВЛЯЕТСЯ СРАВНЕНИЕ---");
g:=0;
repeat
d:=0;
for f:=0 to 7 do
begin port[888]:=f;
b:=port[889];
if (b and 64)=64 then d:=d+(exp(f*ln(2)));
x:=round(d);
end;
port[888]:=32;
read(a,c);
e:=seekeof(a);
if c<>d then writeln(g,' : " , x," вместо " ,c);
g:=g+1;
until e=true;
writeln;
sound(440);
delay(1000);
nosound;
writeln("ОТКЛЮЧИТЕ +5 В") ;
close(a);
end.
(* COPYRIGHT 1991 Patrick GUEULLE *)
```

```
program VIROM;
uses crt;
var k:integer;
    g:integer;
    f:integer;
    b:integer;
begin
clrscr;
writeln("Объем СППЗУ в Кб?");
readln(k);
k:=(k*1024)-1;
clrscr;
writeln("ВНИМАНИЕ! +5 В ДОЛЖНО БЫТЬ ОТКЛЮЧЕНО");
port[888]:=64;
writeln("подключите тестируемое СППЗУ, затем нажмите ENTER");
readln;
clrscr;
writeln("подайте +5 В, затем нажмите ENTER");
readln;
clrscr;
writeln("--ОСУЩЕСТВЛЯЕТСЯ ТЕСТИРОВАНИЕ НА ЧИСТОТУ--");
for g:=0 to k do
begin
for f.=0 to 7 do
begin port[888]:=f;
b:=port[889];
if (b and 64)<>64 then write("*");
end;
port[888]:=32;
end;
writeln("--- ОТКЛЮЧИТЕ +5 В ---");
end.
(* COPYRIGHT 1991 Patrick GUEULLE *);
```

Программа VEROM выводит на экран только содержимое дефектных адресов, в то время как программа VIROM просто выводит звездочки, число которых определяется процентом недостаточно «чистых» ячеек. Вот почему эти программы работают значительно быстрее LECROM и PROGROM.

ПРОГРАММИРОВАНИЕ И СЧИТЫВАНИЕ ОЗУ ZEROPOWER

Микромодули ОЗУ со встроенной литиевой батарейкой (ZEROPOWER) удобно применять вместо СППЗУ в процессе разработки и отладки.

Отпадает необходимость в использовании ультрафиолетовых лучей, и само программирование осуществляется моментально.

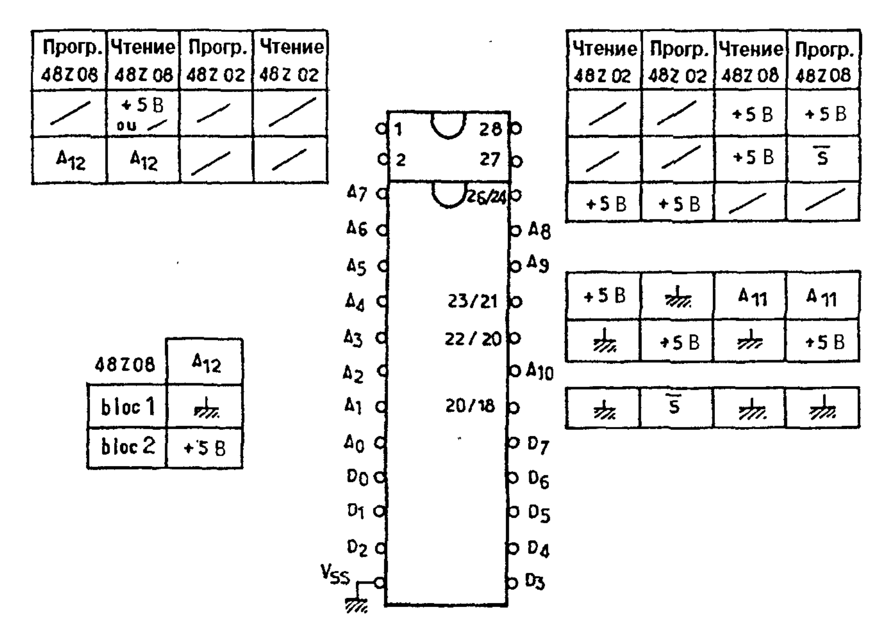


Рис 5.13. Подключение основных микромодулей ZEROPOWER

На рис. 5.13 показаны соединения, которые следует реализовать для программирования и считывания двух наиболее распространенных микромодулей энергонезависимых ОЗУ: МК 48Z02 и МК 48Z08 компании SGS-Thomson, подобных 2716 и 2764.

Процедура чтения аналогична той, что применялась для СППЗУ, и, следовательно, в ней используется та же самая программа LECROM.

Программирование, напротив, требует ПО, названного именно PROGRAM и PRORAM в Basic и Pascal соответственно. (Program – зарезервированное слово в языке Turbo Pascal.)

```
10 REM -- PROGRAM.BAS --
20 CLS:OUT 888,0
30 PRINT"ОТКЛЮЧИТЕ ПИТАНИЕ"
40 PRINT"имя файла, записываемого в ОЗУ с батарейкой?"
50 INPUT F$;F$=F$+" ROM"
60 OPEN"i",#1,F$
70 DIM M(4096).CLS
80 PRINT:PRINT"--ОСУЩЕСТВЛЯЕТСЯ СЧИТЫВАНИЕ ФАЙЛА--"
90 F=0
100 IF EOF(1) THEN 140
110 INPUT#1,M(F)
```

```
120 F=F+1
130 GOTO 100
140 CLOSE#1:CLS
150 PRINT"подключите ОЗУ с батарейкой"
160 PRINT"минимум на";F;"байтов"
170 PRINT"затем нажмите ENTER"
180 INPUT Z$;CLS
190 PRINT"подайте +5 В, затем нажмите ENTER"
200 INPUT Z$:CLS
210 PRINT"--ОСУЩЕСТВЛЯЕТСЯ ПРОГРАММИРОВАНИЕ--"
220 FOR G=0 TO F-1
230 D=M(G)
240 OUT 888,D
250 OUT 890.1
260 OUT 890,0
270 NEXT G
280 PRINT:PRINT:PRINT·BEEP:CLS
290 PRINT"ОТКЛЮЧИТЕ +5 В, затем нажмите ENTER"
300 INPUT Z$:CLS
310 PRINT"ВЫНЬТЕ ОЗУ":END
320 REM(c)1991 Patrick GUEULLE
```

```
program PRORAM;
uses crt;
var m:array[0..4095] of integer;
    h:string[12],
    f:integer;
    e:boolean;
    g:integer;
    a:text;
begin
clrscr;
port[888]:=0;
writeln("внимание! +5 в должно быть отключено");
writeln;
writeln("имя файла, записываемого в ОЗУ с батарейкой?");
readln(h);
h:=h+'.rom';
assign(a,h);
reset(a);
clrscr;
writeln("--ОСУЩЕСТВЛЯЕТСЯ СЧИТЫВАНИЕ ФАЙЛА--");
f:=0;
repeat
read(a,m[f]);
e:=seekeof(a);
f:=f+1;
until e=true;
```

```
close(a);
clrscr,
writeln("ПОДКЛЮЧИТЕ ОЗУ С БАТАРЕЙКОЙ"),
writeln("минимум на",f,' байтов "),
writeln("затем нажмите ENTER");
readln,
clrscr;
writeln("ПОДАЙТЕ +5 В, затем нажмите ENTER"),
readln;
clrscr;
writeln("--ОСУЩЕСТВЛЯЕТСЯ ПРОГРАММИРОВАНИЕ--");
for g:=0 to f-1 do
begin
port[888]:=m[g];
port[890]:=1;
port[890]:=0;
end;
port[888]:=0;
clrscr;
writeln("ОТКЛЮЧИТЕ +5 В, затем нажмите ENTER");
readln,
clrscr;
writeln("ВЫНЬТЕ ОЗУ");
end.
(* COPYRIGHT 1991 Patrick GUEULLE *)
```

Однако исполняемый файл, полученный в результате компиляции одной из этих двух версий, можно переименовать в PROGRAM.EXE, что и было сделано.

ИСТОЧНИК ПИТАНИЯ ДЛЯ ПРОГРАММИРОВАНИЯ СППЗУ

При программировании запоминающих устройств типа СППЗУ требуется несколько специальных напряжений питания 5 В (обычно V_{CC} или V_{DD}) и V_{PP}, величина которого у разных типов устройств бывает различной.

Еще несколько лет назад большая часть СППЗУ программировалась при подаче напряжения V_{PP} = +25 В. Потом появились микросхемы, для которых было достаточно напряжения +21 В или даже +12,5 В, причем наименование микросхемы иногда оставалось прежним! Итак, нужно знать, что рекомендованное значение V_{PP} требуется соблюдать в пределах около полувольта, иначе вы рискуете неудачно провести программирование или «убить» СППЗУ, но в любом случае – сократить срок его службы.

Кроме того, напряжение V_{PP} ни в коем случае не должно быть подано на запоминающее устройство, если в этот момент даже на долю

секунды будет отсутствовать питание +5 В: это напряжение следует подать и снять до отключения главного питания, иначе вы просто разрушите СППЗУ.

Желательно организовать систему защиты так, чтобы V_{PP} спадало быстрее, чем V_{CC}, и дольше бы восстанавливалось, когда снова появится сетевое напряжение. Этот цикл, кроме того, должен соблюдаться при любых токах потребления, соответственно по цепям V_{CC} и V_{PP} (в этом процессе существуют заметные отличия между СППЗУ разных типов и даже изготовителей).

Схема, представленная на рис. 5.14, выполнена в полном соответствии с техническим заданием, которое только что было определено, и при этом изготовлена из минимально возможного количества радиодеталей. Очевидно, можно сделать еще лучше, например с помощью интегральных стабилизаторов напряжения, работающих в следящем режиме, но это совсем необязательно.

Выпрямитель типа удвоителя напряжения позволяет получить от простого трансформатора 2×9 В (или 18 В) требуемое нестабилизированное напряжение, величина которого достаточно высока (около 50 В). Эта конфигурация к тому же позволяет довольствоваться не слишком громоздкими фильтрующими конденсаторами емкостью 2200 мкФ × 25 В.

С первого проходного транзистора BD 135, смещение на который подается со стабилитрона на 5,6 В, снимается нужное напряжение +5 В с максимальным током нагрузки около 100 мА. При уменьшении входного напряжения примерно на 45 В этот транзистор будет сильно нагреваться, поэтому следует использовать более мощный радиатор.

Это нужно для того, чтобы стабилизатор располагал солидным запасом энергии, накопленной в фильтрующих конденсаторах. В случае отключения сетевого питания напряжение +5 В начнет спадать только тогда, когда нестабилизированное напряжение будет равно примерно 7 В. В течение этого времени напряжение V_{PP} уже заметно уменьшится.

Эта первая «линия защиты» дублируется второй системой безопасности – стабилитроном на 8,2 В, включенным последовательно с верхним плечом цепи смещения проходного транзистора в цепи V_{PP} (2N1890 или подобный с максимальным рабочим напряжением как минимум 50 В). Благодаря резистору сопротивлением 10 кОм, который включен между земляной шиной и базой этого транзистора, он будет закрываться не позднее того момента, когда нестабилизированное напряжение на

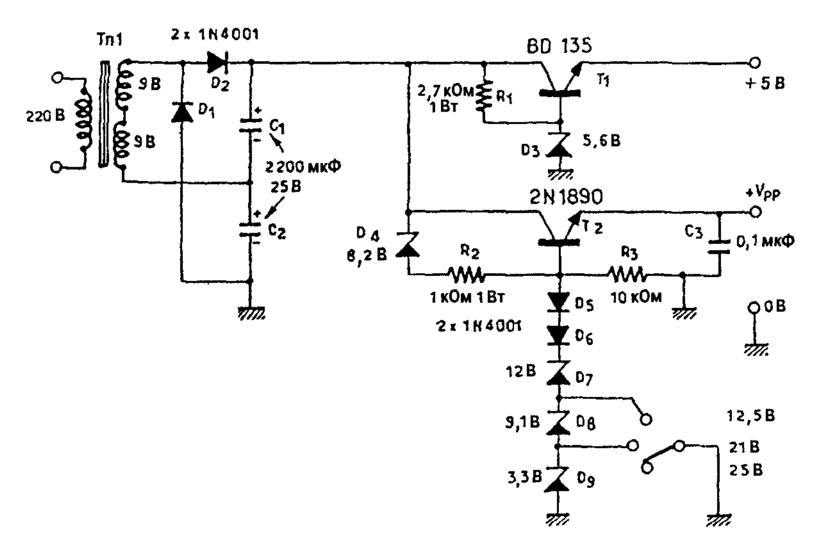


Рис. 5.14. Схема источника питания

входе уменьшится до 8,2 В, или немного раньше. Но мы знаем, что в этот момент напряжение +5 В все еще в норме. И наоборот, при подключении или восстановлении сетевого питания напряжение V_{PP} появится только тогда, когда основное питание +5 В будет уже в норме.

Выбор нужного значения V_{PP} из трех «самых классических» осуществляется частичным закорачиванием цепочки из стабилитронов и кремниевых диодов, подобранных так, чтобы максимально приблизиться к рекомендуемому значению напряжения и как можно сильнее ослабить дрейф напряжения с ростом температуры (напомним, что температурные коэффициенты напряжения стабилитронов и обычных диодов противоположны).

Очевидно, можно составить и другие комбинации в зависимости от определенных потребностей и возможностей каждого.

Считается, что работа с выключателем более удобна и более надежна, чем регулировка потенциометром по показаниям вольтметра, но ошибиться в положении переключателя можно и в подобном случае! Часто на СППЗУ, требующие напряжения V_{PP}, отличного от 25 В, наносят особенную маркировку, но не лишним будет обратиться к каталогу производителя, особенно в тех случаях, когда это касается микросхем, появившихся совсем недавно (в частности, КМОП).

На печатной плате, топология которой показана на рис. 5.15, размещаются (согласно рис. 5.16) все радиодетали схемы и сетевой трансформатор, предназначенные для него контактные площадки, вероятно, надо будет переразвести в соответствии с цоколевкой того трансформатора, который у вас есть, при условии, что вы не предпочитаете расположить его отдельно.

Транзистор 2N1890, так же как и BD135, должен интенсивно охлаждаться, хотя и нагревается он, очевидно, слабее. Для V_{pp} = 12,5 В

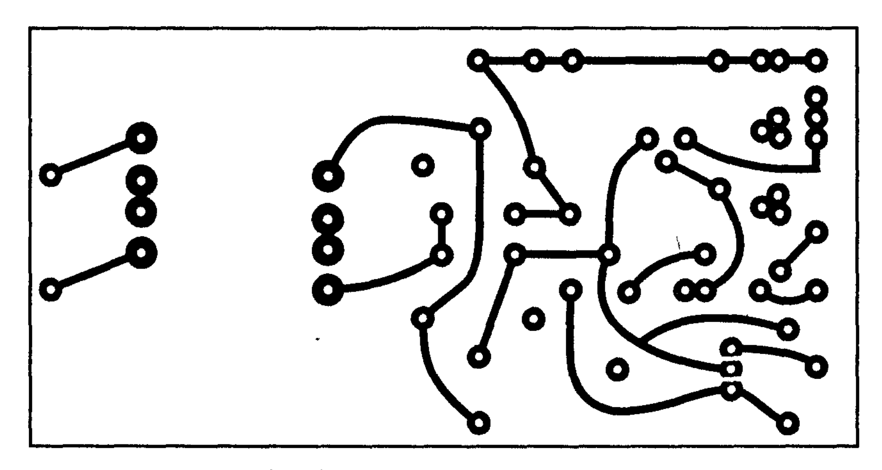


Рис. 5.15. Топология печатной платы

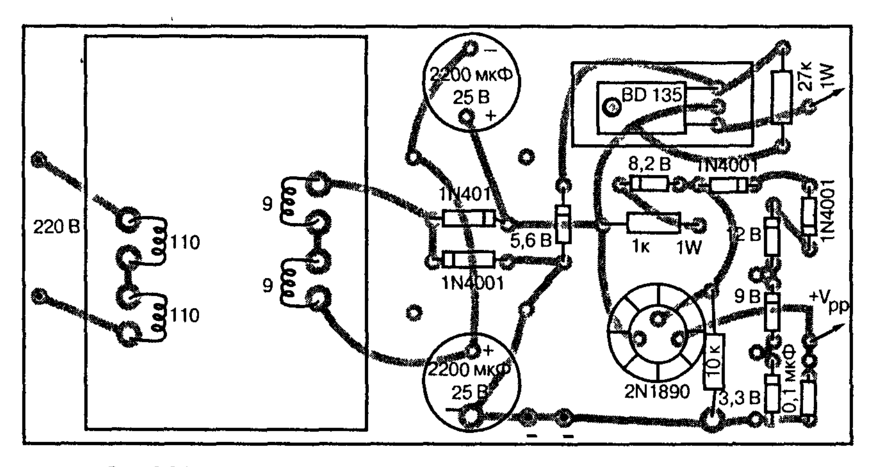


Рис. 5.16. Схема размещения элементов на плате источника питания

Таблица к рис. 5.16

Перечень элементов		
Наименование	**Тип/Номинал**	**Примечание**
Резисторы	1 кОм	1 Вт
	2,7 кОм	1 Вт
	10 кОм	
Конденсаторы	2200 мкФ / 25 В	2 шт.
	0,1 мкФ	
Диоды	1N 4001	4 шт.
Стабилитроны	9,1 В	
	3,3 В	
	8,2 В	
	5,6 В	
	12 В	
Транзисторы	BD 135	
	2N 1890	
Прочее	Трансформатор 220 В / 18 В или 220 В / 29 В	
	Шнур для подключения сетевого питания	
	Переключатель на 3 положения	

он должен, несмотря ни на что, понизить напряжение больше чем на 38 В, что при токе 80 мА соответствует рассеиваемой мощности около 3 Вт! Для 25 В, напротив, можно получить до 110 мА, не превышая то значение, которое соответствует максимально допустимой рассеиваемой мощности прибора 2N1890.

Если необходимы большие токи (например, для одновременного программирования нескольких запоминающих устройств, включенных параллельно), можно легко найти транзисторы с большими максимально допустимыми токами, но тогда надо будет позаботиться о том, чтобы их коэффициенты усиления оставались приемлемыми. Предпочтительнее выбирать составные транзисторы (Дарлингтона).

Разумеется, очень важно, чтобы трансформатор отдавал необходимый ток. Для предлагаемой базовой схемы вполне достаточно модели с габаритной мощностью 8–10 ВА. После сборки эту плату можно разместить в корпусе программатора, в котором остается много свободного места, или в отдельном корпусе. В обоих случаях, но особенно в первом, следует позаботиться об изоляции той части, которая

гальванически связана с сетью 220 В. При необходимости в эту цепь добавляют выключатель и плавкий предохранитель, а при желании – и контрольный индикатор сети.

ПРОГРАММИРОВАНИЕ ispGAL22V10

Обычная GAL22V10 программируется, а ispGAL22V10 компании Lattice «загружается». Это происходит потому, что программирование радиоэлементов требует, как мы видели, подачи на определенное время некоторых напряжений, прикладываемых по очереди к большому количеству выводов.

Напротив, загрузка заключается в последовательном диалоге между радиоэлементом и ПК по кабелю с очень небольшим числом линий и только под напряжением питания +5 В. При этом весь сложный процесс происходит автоматически внутри радиоэлемента, причем совершенно «прозрачным» для оператора способом.

В то время как классические программаторы воздействуют на стандартные выводы радиоэлемента, загрузка может производиться посредством выводов, предназначенных исключительно для этого. Поэтому программирование или перепрограммирование радиоэлемента становится возможным, даже если он уже установлен внутри самой схемы пользователя, и это при том, что вся схема находится под напряжением! Фирма Lattice называет такой процесс In Situ Programmability (в переводе с англ. – программирование на месте). Но вопреки загружаемым радиоэлементам, базирующимся на ОЗУ, микросхема ispGAL22V10 сохраняет записанные данные до тех пор, пока они не будут сознательно стерты. Однажды запрограммированная, эта 22V10, как и другие, при необходимости может быть изъята из одной платы и вставлена в другую без потери той «схемы», которую она содержит. Это особенно ценно при разработке опытных образцов.

Но как к обычной 22V10 добавить выводы загрузки, полностью поддерживая при этом совместимость на уровне цоколевки с обычными моделями? Просто ограничиться корпусом PLCC с 28 выводами, у которого четыре вывода не используются. Получается цоколевка, изображенная на рис. 5.17, где показаны линии загрузки SDO, SDI, MODE и SCLK. Разумеется, это неприемлемо для корпуса DIP с 24 выводами, в котором также размещают GAL22V10, но при этом вы всегда можете использовать адаптер PLCC-DIP.

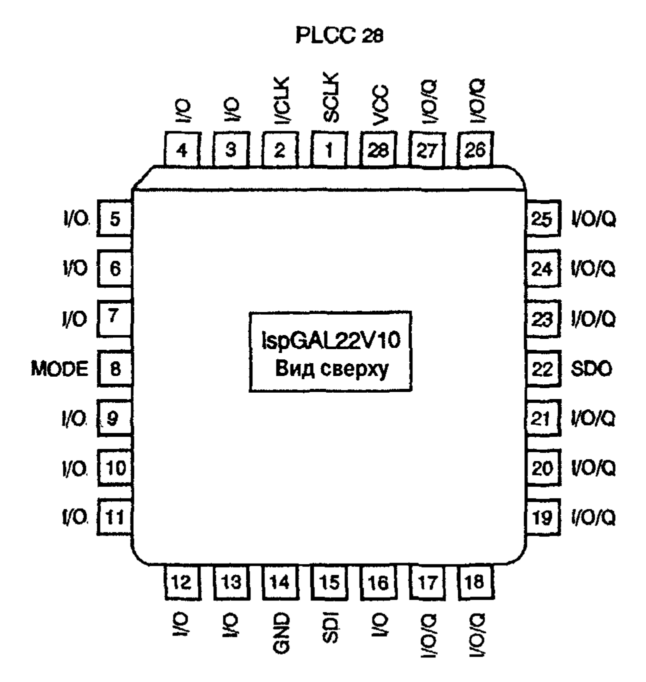


Рис. 5.17. Цоколевка ispGAL22V10 (корпус PLCC 28, вид сверху)

Рассмотрим последовательно два подхода, позволяющих программировать ispGAL22V10 посредством совместимого ПК:

- «чистое» и простое использование – Starter Kit, продаваемый компанией Lattice, который стоит приобрести, если вы решили продолжить эксперименты со всем семейством ispLSI;
- сборка «домашнего» программатора, используемого с программным обеспечением, предложенным компанией Lattice (см. главу 6).

Чтобы воплотить в жизнь первое решение, необходимы соединительный кабель и программное обеспечение.

Хотя спецификация компании Lattice содержит все детали протокола загрузки, начиная с файла JEDEC, полученного от абсолютно любого логического компилятора (например, Prologic), по-настоящему интересно писать свое собственное ПО только в том случае, если необходимо, чтобы само устройство, содержащее микоросхемы GAL,

было бы в состоянии их перепрограммировать (для облегчения разработки таких модулей программирования предлагается целая библиотека на языке С). В большинстве остальных случаев используют системы разработки для операционной системы Windows, предлагаемые для этой цели компанией Lattice. Кабели для загрузки, входящие в комплект, снабжены модульными разъемами с восемью контактами (RJ-45) и подключаются к параллельному порту LPT1 через адаптер DB25.

Отметим, что этот адаптер содержит небольшую электронную схему, основное назначение которой – буферизация линий данных и подача на схему адаптера напряжения питания +5 В с той платы, на которой размещена программируемая 22V10. Другой конец первого кабеля также снабжен этим модульным разъемом, а второй кабель – соединителем AMP с шагом контактов 2,54 мм. В обоих случаях соединение с выводами 22V10 осуществляет пользователь, который должен установить специальный разъем на печатной плате самого устройства.

Во всех этих кабелях, разъемах и прочих переходниках, представленных на рис. 5.19, к сожалению, не предусмотрены возможности программирования самой 22V10 (просто микросхемы, вне платы и без обвязки) в лабораторных или домашних условиях. Но мы собираемся устранить этот недостаток.

Решение проблемы состоит в том, чтобы соединить с панелькой PLCC28, цоколевка которой изображена на рис. 5.18, шесть контактов разъема AMP для кабеля загрузки с одной ее стороны, а с другой – питание +5 В.

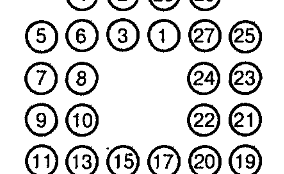


Рис. 5.18. Цоколевка панельки для PLCC28 (вид сверху)

Именно для этой цели предназначена маленькая печатная плата, топология которой представлена на рис. 5.20 и которую надо собрать согласно схеме размещения элементов (рис. 5.21). Внешний вид всего устройства показан на рис. 5.22.

Соединитель выполнен из куска разрезной колодки с квадратными штырьками. Один из штырьков находится напротив ключа на разъеме, оснащающем кабель, и его следует извлечь или отрезать. Но проще из любых деталей собрать маленький программатор, соединяющийся непосредственно с параллельным портом ПК, чем использовать кабели из набора для начинающих. Это будет

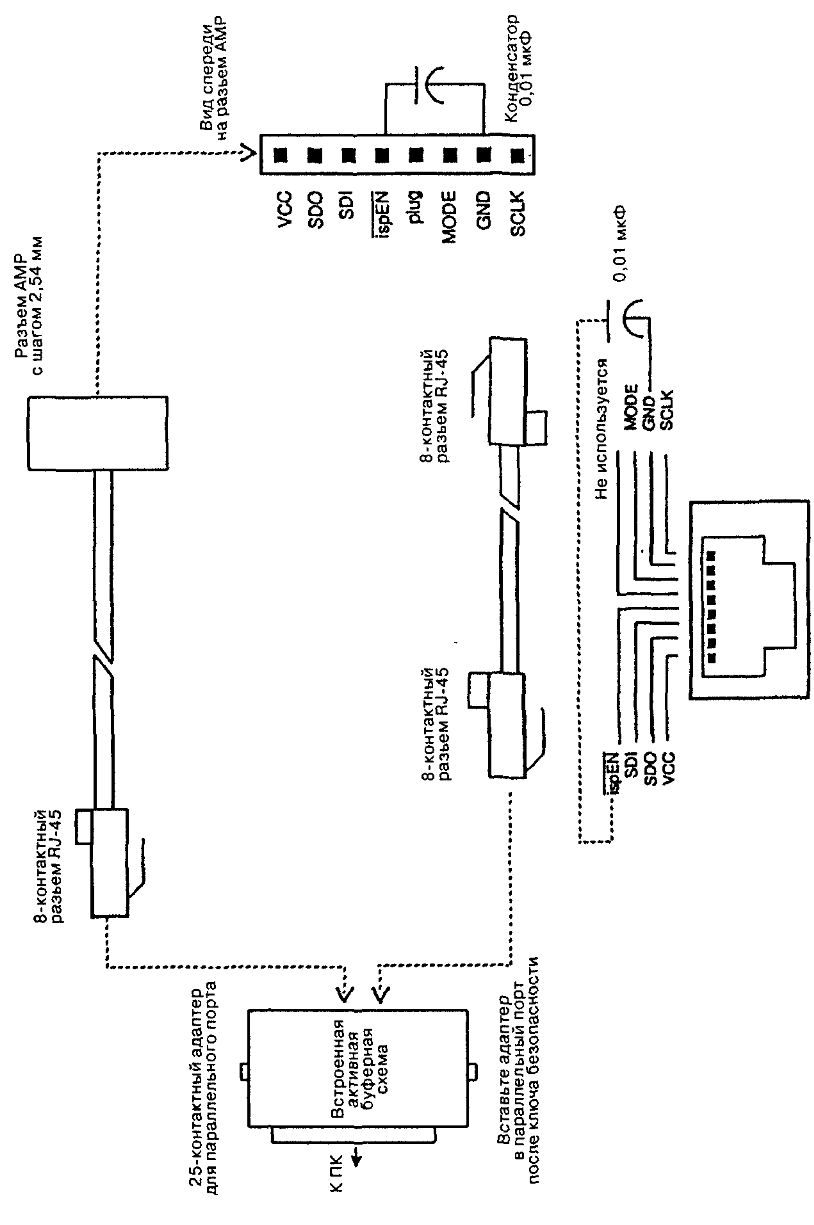


Рис. 5.19. Подключение кабелей из стартового набора Lattice

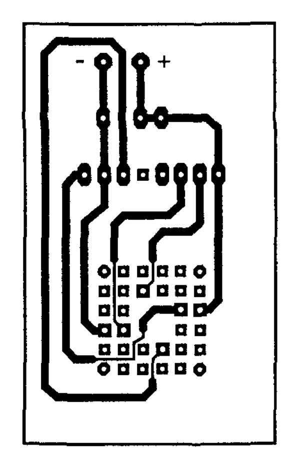


Рис. 5.20. Печатная плата адаптера PLCC28

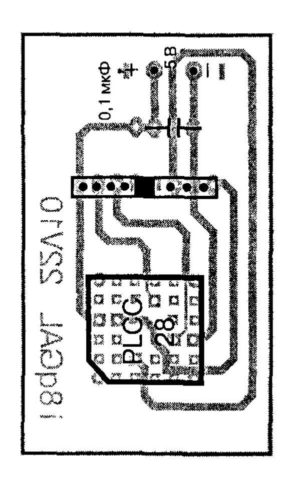


Рис. 5.21. Схема размещения элеметов адаптера PLCC28

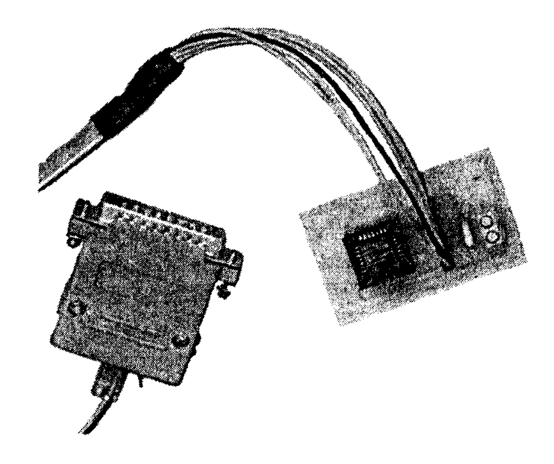


Рис. 5.22. Внешний вид собранного приспособления

хорошим решением для читателей, которые хотели бы ограничиться использованием программного обеспечения, рассматриваемого на страницах этой книги, и обойтись без дополнительных капиталовложений.

Схема, представленная на рис. 5.23, разработана с учетом рекомендаций фирмы Lattice, но для ее изготовления требуется более распространенная интегральная микросхема.

На печатной плате, топология которой представлена на рис. 5.24, устанавливаются все необходимые радиодетали, что приводит к минимально возможной длине всех соединений.

По завершении монтажа, выполняемого согласно схеме размещения элементов на рис. 5.25 (не забывая о трех перемычках!), достаточно

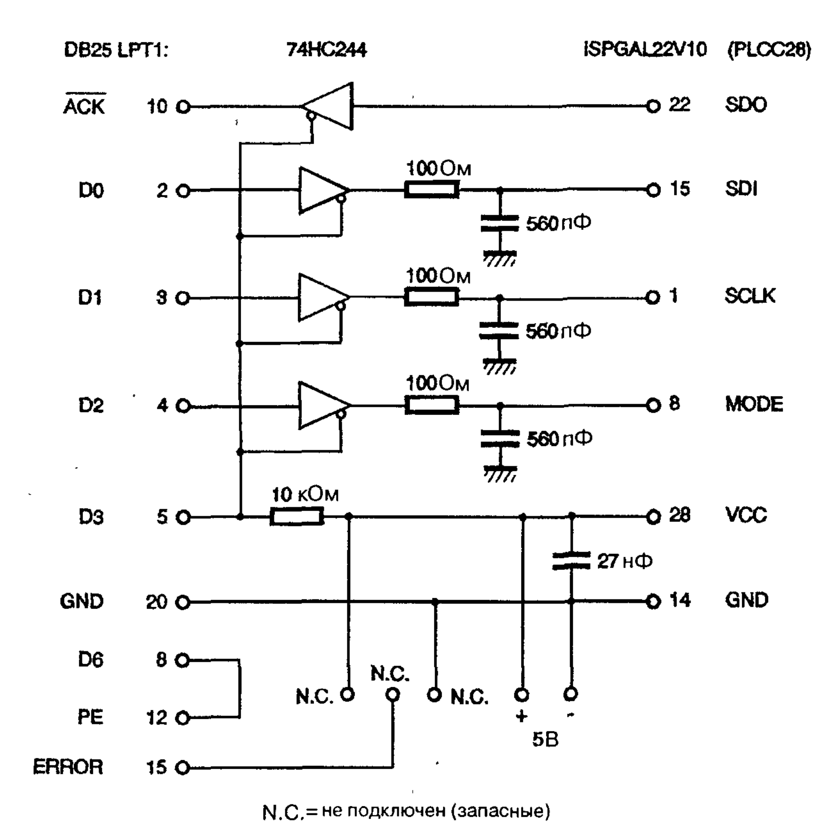


Рис. 5 23. Схема программатора ispGAL22V10

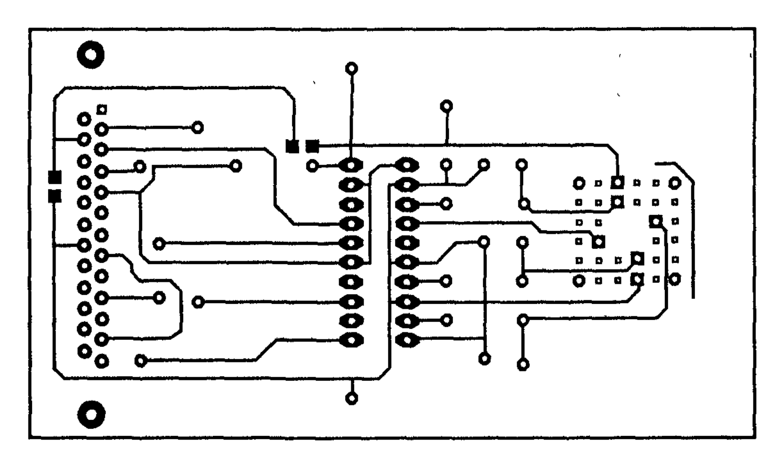


Рис. 5.24. Топология печатной платы программатора

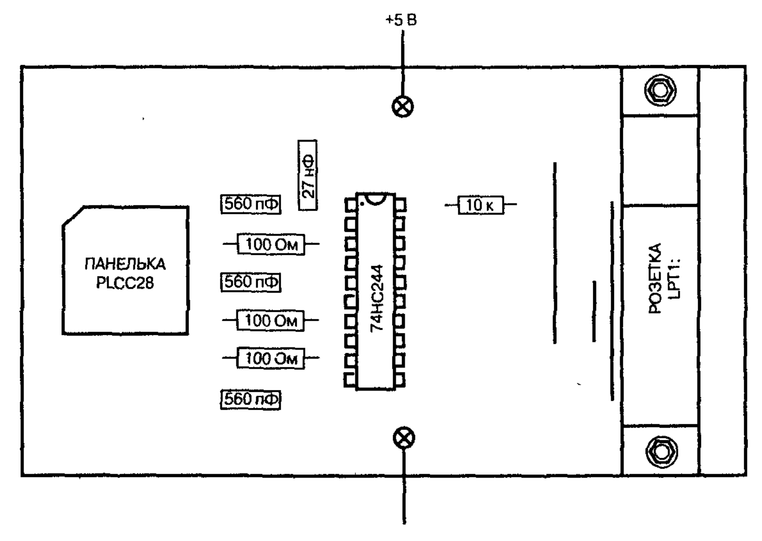


Рис. 5.25. Схема размещения элементов программатора

подать на плату питание +5 В (например, с контакта 1 разъема типа DB15 игрового порта ПК) и подсоединить очень короткий кабель с двумя разъемами типа DB25 (вилки), один из которых включается в программатор, а другой – в параллельный порт LPT1 того же ПК. Разумеется, использование этого программатора идентично использованию штатного кабеля загрузки. Внешний вид программатора показан на рис. 5.26.

В обоих случаях очень важно правильно различать нумерацию выводов у 22V10, размещенных в корпусах DIP24 и PLCC28, цоколевка которых приведена в табл. 5.1 (за исключением четырех выводов загрузки, отмеченных на рис. 5.17). Имена сигналов, используемые

Таблица 5 1. Соответствие цоколевок корпусов DIP24 и PLCC28

Контакты DIP24	Контакты PLCC28	Назначение
1	2	Ввод/синхронизация
2	3	Ввод
3	4	Ввод
4	5	Ввод
5	6	Ввод
6	7	Ввод
7	9	Ввод
8	10	Ввод
9	11	Ввод
10	12	Ввод
11	13	Ввод
12	14	Общий
13	16	Ввод
14	17	Ввод/вывод
15	18	Ввод/вывод
16	19	Ввод/вывод
17	20	Ввод/вывод
18	21	Ввод/вывод
19	23	Ввод/вывод
20	24	Ввод/вывод
21	25	Ввод/вывод
22	26	Ввод/вывод
23	27	Ввод/вывод
24	28	VCC

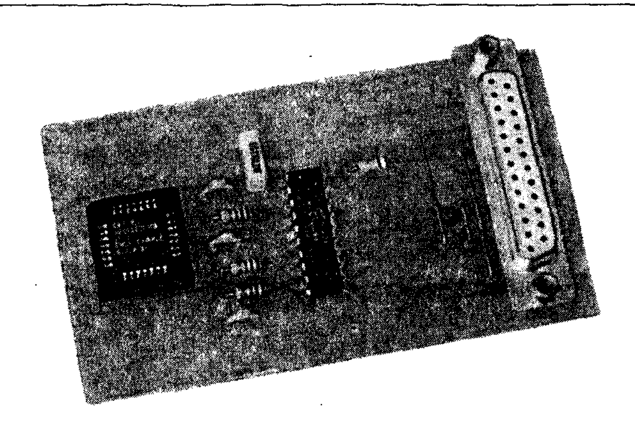


Рис. 5.26. Внешний вид программатора

логическим компилятором Prologic, определены для номеров выводов корпуса DIP. Когда вы будете писать исходный код для приложения, ориентированного на применение ispGAL22V10 в корпусе PLCC, старайтесь избегать любой неточности.

ПРОГРАММИРОВАНИЕ ispLSI 1016 И 2032

Starter Kit компании Lattice содержит ПО – PDS, необходимое для удобства работы с ispLSI 1016 и 2032, программирование которых (или перепрограммирование после очищения) требует только сборки маленького адаптера, снабженного подобранной панелькой PLCC.

На рис. 5.27 представлена разводка простой печатной платы, на рис. 5.28 – соответствующая схема размещения элементов, а на рис. 5.29 – внешний вид адаптера.

Следует соблюдать правильную ориентацию панельки PLCC44 (вариант для монтажа в отверстия платы, а не SMD), определяемую срезанным уголком, и создать ключ, изъяв соответствующий контакт из разрезной колодки с квадратными штырьками, которая служит ответной частью для разъема кабеля.

Чтобы использовать этот прибор (совместимый с ispLSI1016 и ispLSI2032), достаточно подключить кабель загрузки (только после установки микросхемы!) и питание 5 В, которое можно просто снимать с контакта 1 разъема игрового порта ПК.

Все остальное – программирование, проверка и стирание – происходит при обращении к соответствующему пункту меню программы PDS (см. рис. 5.30) или к ПО (см. главу 6), которое предназначено для программирования ispGAL22V10 под Windows.

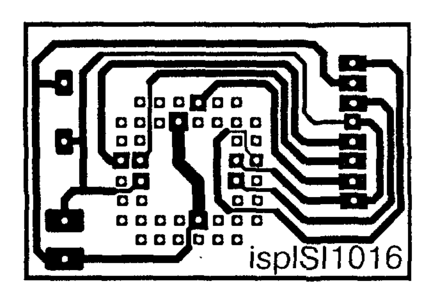


Рис. 5.27. Печатная плата адаптера PLCC44

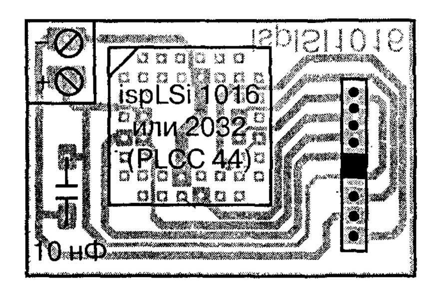


Рис. 5.28. Схема размещения элементов на плате адаптера PLCC44

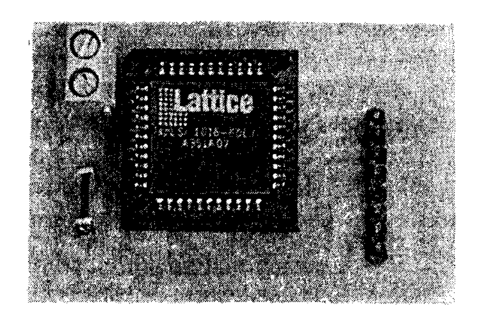


Рис. 5.29. Внешний вид адаптера

ПРОГРАММАТОР МИКРОКОНТРОЛЛЕРОВ PIC

Даже в составе набора для начинающих Starter Kit компании Microchip промышленный програматор PIC обходится значительно дороже, чем

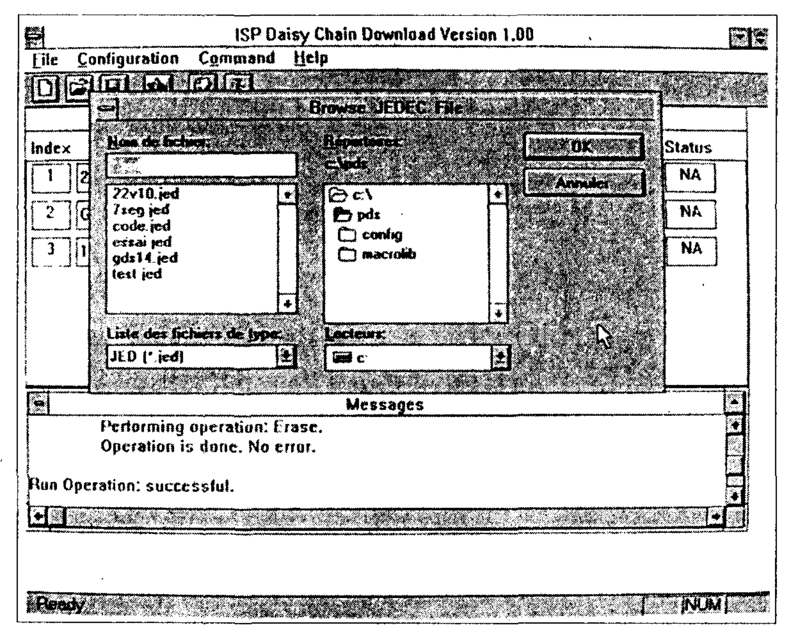


Рис. 5.30. Выбор файла для загрузки в PDS (фотография экрана)

самостоятельная его сборка! При условии использования последовательного способа программирования, доступного с момента появления новых моделей PIC (в частности, 16C84), можно собрать программатор, удивительно простой и полностью совместимый с мощным ПО.

А знали ли вы то, что многие микросхемы семейства PIC разработаны на основе концепции, сходной с концепцией isp, которая подразумевает программирование их (или перепрограммирование) непосредственно на плате, где они находятся, причем даже без отключения питания?

Переход в режим программирования происходит при подаче на вывод сброса ($\overline{MCLR}$) высокого напряжения, которое в данном случае находится в диапазоне от 12 до 14 В. С этого момента вывод RB6 становится входом сигнала синхронизации (CLOCK), а вывод RB7 – линией ввода-вывода данных (DATA).

Выводы V_{dd} и V_{ss} сохраняют свое обычное назначение – питание PIC его номинальным напряжением 5 В, в то время как все другие выводы становятся недействующими. Тогда можно программировать,

читать или стирать PIC, просто ведя с ним последовательный диалог по линии DATA.

Правила этого диалога полностью раскрыты в документе объемом до десяти страниц (PIC16C84 EEPROM Memory Programming Specification – спецификации для программирования PIC16C84 с памятью ЭСППЗУ), который можно получить у Microchip или обратиться к компакт-диску, распростряемому этим производителем. Тем не менее допускается существенно смягчить эти строгие правила, если ограничиться задачами разработки опытных или единичных образцов.

Идея реализации схемы чрезвычайно проста: обеспечить PIC напряжением, снимаемым с линии TXD последовательного порта (RS232), и использовать другие линии этого порта для диалога (см. рис. 5.31).

Это позволяет полностью исключить внешние источники питания или батареи, что очень удобно на практике.

Если это положение не вызывает проблем для получения 5 В с помощью интегрального стабилизатора 78L05, необходимый минимум в 12 В для вывода $\overline{\text{MCLR}}$ предполагает, напротив, что последовательный порт компьютера является «настоящим» RS232C, и рабочие уровни у него не меньше ±12 В.

Поэтому программатор заведомо не сможет фукционировать на ПК, оснащенных последовательными портами, которые работают при напряжениях ±5 В. Не следует доверять, в частности, ноутбукам.

В подобных случаях выход из создашегося положения состоит в том, чтобы добавить последовательный порт с нужными свойствами, например в виде платы multi I/O – дешевого и, кроме того, полезного приспособления.

Если напряжение 12 В, как положительное, так и отрицательное, присутствует на линиях последовательного порта, нам надо ограничить его уровнями безобидными для PIC, что и сделано с помощью нескольких стабилитронов, которые, впрочем, работают в обоих направлениях.

После изготовления маленькой печатной платы в соответствии с топологией, показанной на рис. 5.32, производят монтаж всех радиодеталей согласно схеме размещения элементов (рис. 5.33), тщательно соблюдая при этом ориентацию полярных элементов.

Не следует использовать стабилитроны на напряжение ниже 5,6 В, так как в этом случае схема будет неработоспособна. Подключение

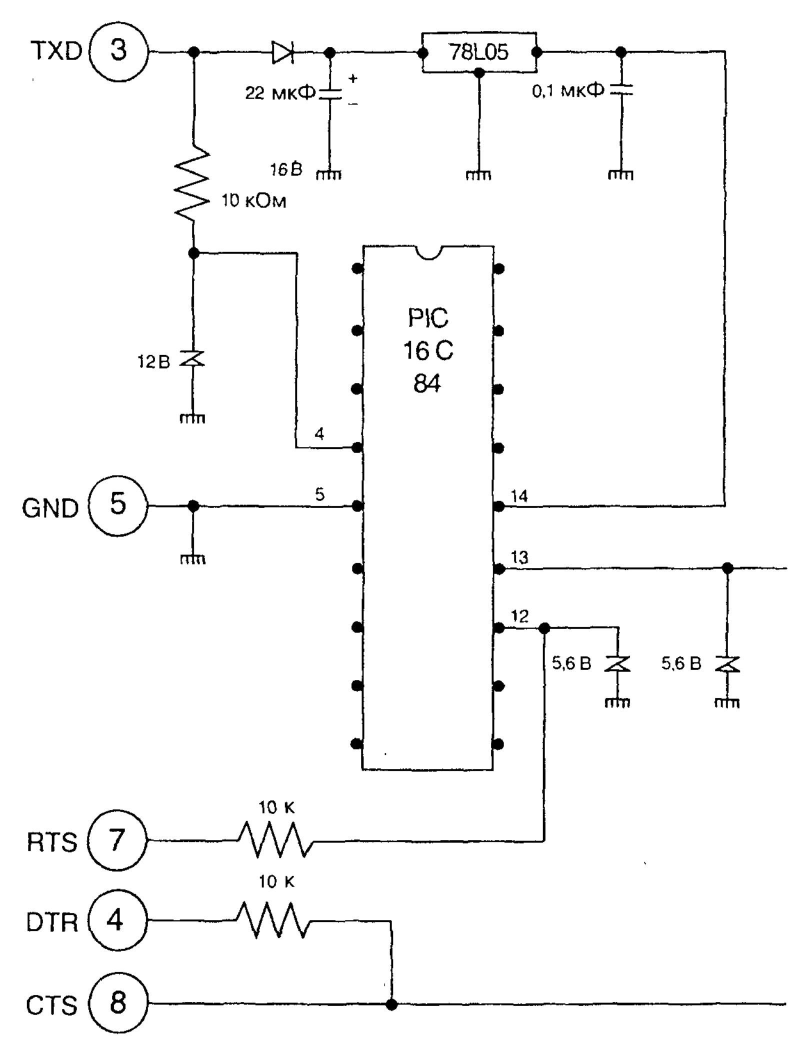


Рис 5.31. Принципиальная схема программатора PIC

к последовательному порту ПК производится либо путем установки платы в разъем, либо с помощью кабеля DB9 розетка-вилка (простого удлинителя для последовательного порта). При необходимости можно подключить его через адаптер DB9 – DB25, но ни в коем случае не перекрещенным кабелем (называемым нуль-модемным). Подключение напрямую обязательно!

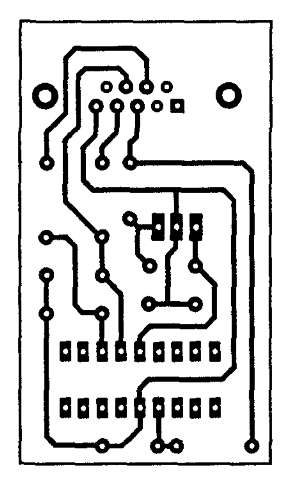


Рис 5.32. Топология печатной платы программатора PIC

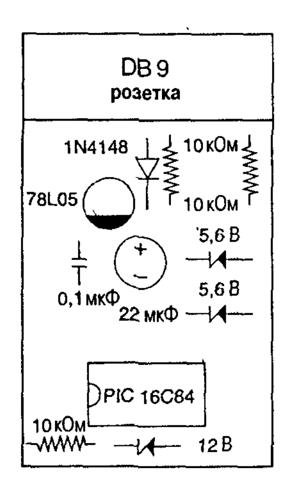


Рис. 5.33. Схема размещения элементов на плате программатора PIC

Таблица к рис. 5.33

Перечень элементов		
Наименование	**Тип/Номинал**	**Примечание**
Резисторы	10 кОм / 0,25 Вт	2 шт.
Конденсаторы	22 мкФ / 16 В	Радиальный
	0,1 мкФ	
Диоды	1N 4148	
Стабилитроны	4,7 В / 0,25 Вт	2 шт.
Микросхемы	78L05	
Прочее	Панелька с 28 гиперболическими контактами	
	Разъем DB9 розетка угловая	

Программное обеспечение для программирования

Прежде чем запускать основное программное обеспечение (PIP-02), необходимо установить резидентный драйвер COM84. Синтаксис, который нужно использовать при этом, следующий:

COM84 COMn (n – номер последовательного порта, от 1 до 4).

Операции можно автоматизировать, написав маленький пакетный файл (названный, например, PIP.BAT), содержание которого может быть таким (для последовательного порта COM1):

```
COM84 COM1
PIP-02
COM84 REMOVE
```

Последняя строка предназначена для того, чтобы выгрузить из памяти драйвер после выхода из программы. Это необходимо во избежание конфликтов с другим программным обеспечением.

Рабочий интерфейс программы PIP-02 (см. рис. 5.34) удобен в использовании. Здесь доступны многочисленные меню, с которыми надо знакомиться постепенно. Не забывайте указывать тип микроконтроллера PIC, предназначенного для программирования (или считывания), прежде чем выполнять какую-либо операцию. В настоящее время программой может быть считан, запрограммирован или стерт только микроконтроллер PIC16C84, хотя допускается (не ручаясь за достоверность) считывать (но не программировать) и микроконтроллер PIC16C71.

Разумеется, следует правильно выставить все «плавкие перемычки» (биты конфигурации), прежде чем приступить к программированию.

Отметим, что файл, используемый для программирования, должен быть представлен в формате INHX8M, который, кстати, генерируется большинством средств разработки для микроконтроллеров PIC.

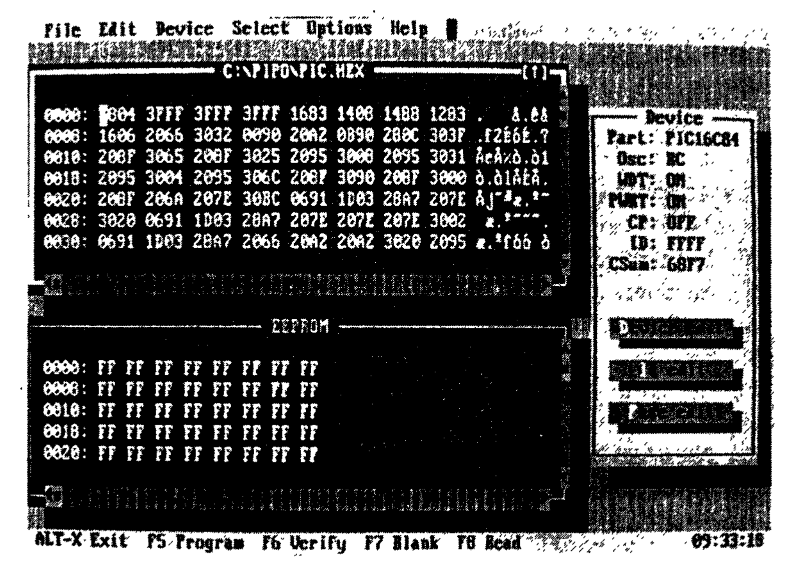


Рис. 5.34. Экран программы PIP-02 (фотография)

И последнее: всегда можно полностью стереть PIC вместе со всеми битами конфигурации, выбрав опцию **Erase** в меню **Device**.

ПРОГРАММИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ СППЗУ

В последнее время все чаще и чаще появляется потребность в считывании, перезаписи или изменении содержания распространенных и дешевых ЭСППЗУ. Простая маленькая схема, приведенная на рис. 5.35, может заменить «универсальный» программатор.

Можно заметить, что эта схема напоминает предыдущую, и не без основания: она действительно работает под управлением того же

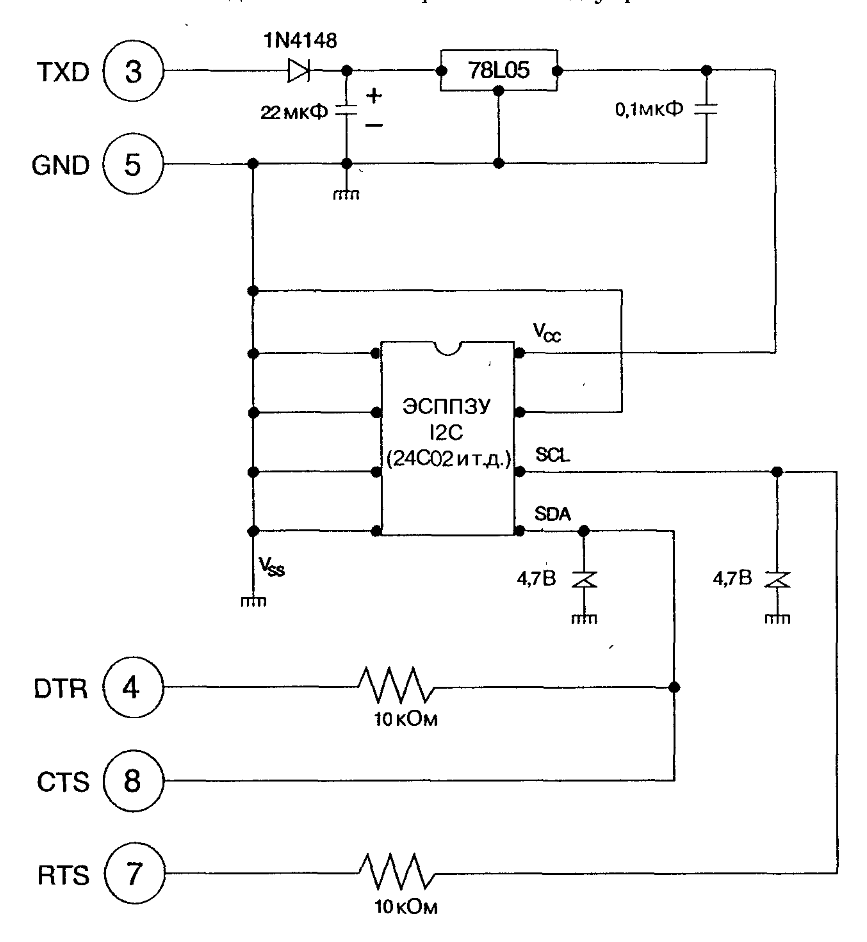


Рис. 5.35. Принципиальная схема программатора последовательных ЭСППЗУ

программного обеспечения (PIP-02), в меню Select которого предлагаются многочисленные микросхемы последовательных ЭСППЗУ, совместимых с протоколом I2C (в основном это 24Cxx).

Уточним, что речь идет о временной версии программы, впоследствии, вероятно, будут поддерживаться и другие семейства запоминающих устройств. Как и PIC, запоминающие устройства I2C связаны с программатором линией данных (SDA) и линией синхронизации (SCL), при этом никакого напряжения питания больше 5 В не требуется. Это доказывает, что ранее оговоренные ограничения, касающиеся совместимости программатора PIC с некоторыми последовательными портами ПК, здесь не имеют значения.

Благодаря интегральному стабилизатору 78L05 и двум стабилитронам на 4,7 В ни на один из выводов микросхемы запоминающего устройства не сможет попасть никакой отрицательный или положительный потенциал больше 5 В, даже при подключении к порту RS232, работающему, как ему и положено, с напряжениями ±12 В. По соображениям защиты от электростатических разрядов желательно вставлять и извлекать микросхему памяти только в том случае, когда программатор не присоединен к ПК.

Таким образом, схему к последовательному порту подключают перед тем, как начать программирование или чтение, и отключают,

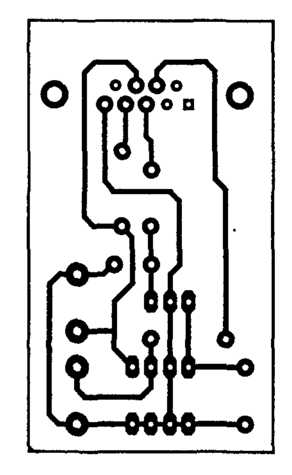


Рис. 5.36. Топология печатной платы программатора последовательных ЭСППЗУ

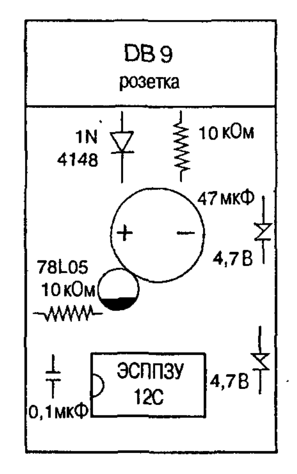


Рис. 5.37. Схема размещения элементов на плате программатора последовательных ЭСППЗУ

Таблица к рис 5 37

Перечень элементов		
Наименование	**Тип/Номинал**	**Примечание**
Резисторы	10 кОм / 0,25 Вт	2 шт.
Конденсаторы	22 мкФ / 16 В	Радиальный
	0,1 мкФ	
Диоды	1N-4148	
Стабилитроны	4,7 В / 0,25 Вт	2 шт.
Микросхемы	78L05	
Прочее	Панелька с 28 гиперболическими контактами	
	Разъем DB9 розетка (угловая)	

соответственно, сразу после этого. Сначала нужно изготовить печатную плату, топология которой показана на рис. 5.36. Эта плата имеет те же размеры, что и плата программатора микроконтроллеров PIC. Следовательно, обе схемы будут вполне взаимозаменяемыми и оснащены одной и той же моделью углового разъема DB9. Размещение нескольких необходимых элементов, которых еще меньше, чем у программатора PIC, производится согласно схеме на рис. 5.37. На рис. 5.38 показан внешний вид программаторов PIC и ЭСППЗУ.

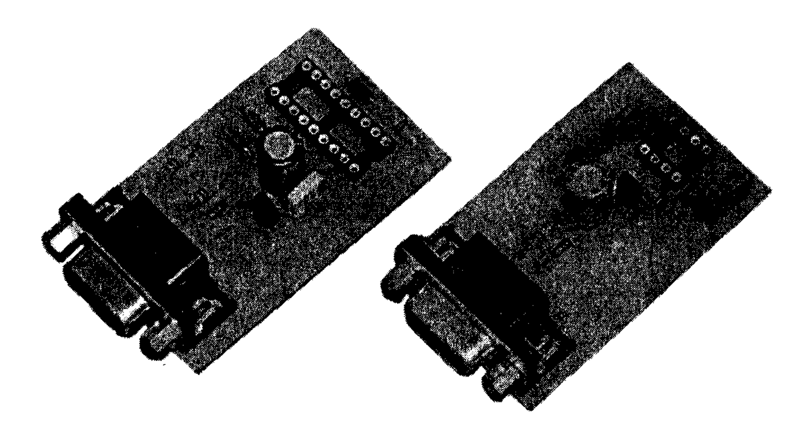


Рис. 5 38. Внешний вид программаторов PIC и ЭСППЗУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПРОГРАММИРОВАНИЯ

Поскольку речь идет о работе с микроконтроллерами PIC или с запоминающими устройствами, то программное обеспечение PIP-02 устанавливатся и запускается одинаково. Непосредственно в пункте **Device** меню **Select** указывают тип запоминающего устройства, которое нужно считать или запрограммировать.

Благодаря файлу инициализации, обновляющемуся во время каждого сеанса работы, PIP-02 «вспоминает» о последнем радиоэлементе, к которому вы обращались, поэтому следует проверить, соответствует ли последняя конфигурация той работе, которую нужно выполнить.

При считывании, как и при программировании, содержание запоминающего устройства передается в виде файла в формате Intel hex (см. рис. 5.39), но можно перекодировать его в любой формат посредством соответствующей утилиты.

С помощью программного обеспечения целиком стереть содержимое ЭСППЗУ нельзя (у этих запоминающих устройств такая операция не предусмотрена). Данный недостаток можно обойти, заранее подготовив файлы .hex, соответствующие «чистым» ЭСППЗУ. Это можно сделать, считав данные из новой микросхемы или же использовав редактор файлов из пакета PIP-02.

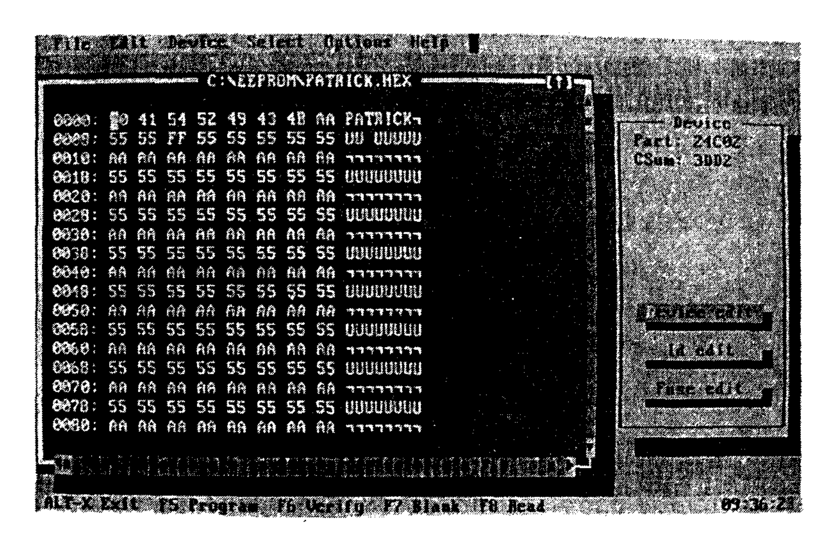


Рис. 5.39. Экран PIP-02 при работе с файлом Iintel hex (фотография)

ГЛАВА СТРАНИЦА

6 ИСПОЛЬЗУЕМОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Все программное обеспечение (его можно копировать для своих собственных нужд, но ни в коем случае не использовать в коммерческих целях), которое вам понадобится при изучении материала, изложенного в этой книге, можно условно разделить на четыре части, а именно:

```
PROLOGIC.EXE 282986
LECROM.PAS   970
PROGRAM.PAS  949
PROGROM.PAS  1554
VEROM.PAS    905
VIROM.PAS    613
LECROM.EXE   7520
PROGRAM.EXE  4048
PROGROM.EXE  6336
VEROM EXE    7456
VIROM EXE    2960
PROGROM.BAS  1254
TRANS.BAS    932
LECROM.BAS   655
VEROM.BAS    673
VIROM.BAS    606
PROGRAM.BAS  834
MQP <DIR>
ISP <DIR>
PIC <DIR>
```

В корневом каталоге содержатся все программы, листинги которых приведены в книге, в версиях на языках GWBasic и TurboPascal.

За исключением программы trans.bas, написанной только на языке Basic, все ПО представлено в виде пар файлов: исходного текста и исполняемого, последний можно запустить из DOS (это касается файлов с расширением .exe).

В принципе именно эти файлы рекомендуется использовать, остальные предназначены в основном для того, чтобы вносить изменения в предложенные программы.

Файл prologic.exe представляет собой самораспаковывающийся архив, в котором собрано все необходимое для работы логического компилятора PROLOGIC. Эта версия компилятора поставляется компанией Texas Instruments для работы с выпускаемыми ею ПЛИС (однако он подходит и для других микросхем)[1].

В подкаталоге mqp содержится демонстрационная версия программного обеспечения английского программатора СППЗУ (©1996,

[1] PROLOGIC – торговая марка компании Inlab Incorporated.

MQP Electronics Ltd.). Его главная ценность – превосходный редактор файлов (PDED), который предлагается как условно бесплатное ПО (shareware). Перед началом работы внимательно прочитайте файл read.me.

Прежде чем приступить к установке ПО, надо переписать содержимое подкаталога mqp на промежуточную дискету. После этого дискету вставить в дисковод А и набрать A:INSTALL.

В подкаталоге isp собраны программы, предлагаемые компанией Lattice для работы с микросхемами ispGAL22V10 и ispLSI (© Lattice Semiconductor Corp.) Их можно установить, запустив в Windows файл setup.exe из каталога isp.

Наконец, подкаталог pic содержит программы для управления программаторами микроконтроллеров PIC16C84 и последовательных ЭСППЗУ, описанными в главе 5.

Достаточно просто скопировать его в какой-нибудь каталог на жестком диске или же на рабочую дискету и использовать непосредственно, если программатор подключен к последовательному порту COM1 (для запуска программы надо набрать на клавиатуре PIP).

ВНИМАНИЕ!

Вы можете стать активным участником процесса книгоиздания в России!

Заполните предлагаемую анкету и укажите, книги на какие темы Вас интересуют.

При полном заполнении анкеты Вы получаете возможность в течение 6 месяцев пользоваться **10% скидкой** при приобретении компьютерной и радиотехнической литературы, имеющейся в нашем Internet-магазине, выпускаемой как «ДМК», так и другими издательствами (всего 700 наименований). Для этого достаточно получить логин и пароль, которые Вы будете использовать при входе в Internet-магазин на сайте www.dmk.ru. Узнать логин и пароль Вы можете, позвонив по тел. 369-33-60 или прислав письмо по электронной почте (info@dmk.ru) после отправки анкеты по адресу: 105023, Москва, пл. Журавлева, д. 2/8, оф. 400.

1. Где Вы приобрели эту книгу? ______________________
город адрес

2. Вы приобрели эту книгу за ____ руб. Это ☐ очень дорого ☐ дороговато ☐ приемлемо

3. Оцените по 5-балльной системе:

	1	2	3	4	5
а) качество выполнения иллюстраций	☐	☐	☐	☐	☐
б) качество изложения материала	☐	☐	☐	☐	☐
в) актуальность рассмотренных тем	☐	☐	☐	☐	☐
г) общее впечатление от книги	☐	☐	☐	☐	☐

4. С какой целью Вы приобрели эту книгу? ☐ для работы ☐ для самостоятельного изучения предмета ☐ для общего развития

5. Если Вы обнаружили какие-либо ошибки или неточности в книге, пожалуйста, перечислите их ниже (с указанием номеров страниц) ______________________

____

____

____

Предлагаем Вам принять участие в подготовке наших будущих изданий. Для этого ответьте на нижеследующие вопросы:

Наиболее актуальные, по Вашему мнению, темы компьютерной и радиотехнической литературы

____

____

____

Основные аспекты, которые, с Вашей точки зрения, должны быть подробно рассмотрены в книге

____

____

____

Примерный объем (*число страниц*) ______________________

Уровень читателя (начинающий, опытный пользователь, ...) ______________________

Приемлемая для Вас цена ________ руб.

Мы приглашаем авторов к сотрудничеству.
Пишите по адресу: books@dmk.ru. Факс: 369-78-74

СДЕЛАЙТЕ ПРИЯТНОЕ ВАШИМ ДРУЗЬЯМ –
ПРИГЛАСИТЕ ИХ С СОБОЙ НА ПРОГУЛКУ В НАШ INTERNET-МАГАЗИН на сайте
www.dmk.ru

017-X Как превратить ПК в универсальный программатор

Патрик Гёлль

Как превратить персональный компьютер в универсальный программатор

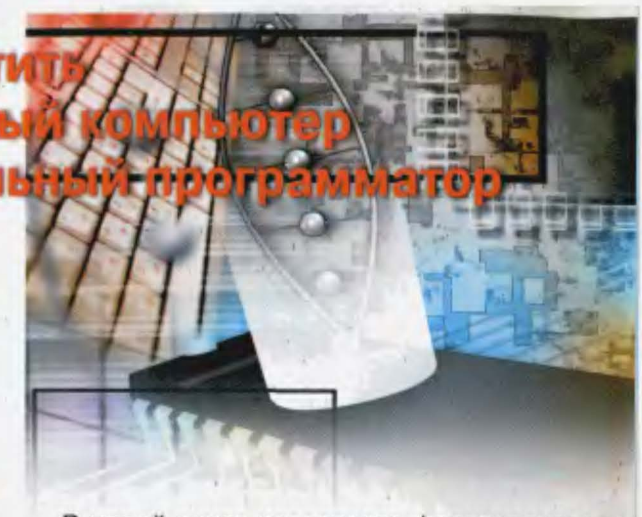
Главный редактор *Захаров И. М.*
Научный редактор *Бряндинский А. Э.*
Литературный редактор *Ноткина А. Б.*
Технический редактор *Прока С. В.*
Верстка *Пискунова Л. П.*
Графика *Понявин С. А.*
Дизайн обложки *Антонов А. И.*

ИД № 00498 от 29.11.99

Подписано в печать 28.04.2000. Формат 60×88 1/16
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 10,5. Тираж 3000 экз. Зак. № 506

Издательство «ДМК», 113184, Москва, Пятницкий пер., д. 3, стр. 3.
Отпечатано в типографии № 9. Волочаевская, 40.

Как превратить персональный компьютер в универсальный программатор





Патрик Гёлль

В новой книге известного французского ин женера и радиолюбителя рассматриваются наи более распространенные типы самых современ ных интегральных микросхем – многократн перепрограммируемых. Представлены все ос новные классы: ИМС памяти, программируемы логические ИМС и микроконтроллеры. Описан простые и надежные программаторы, приведе ны программы для управления ими, рассмотре ны программные комплексы и системы разра ботки для ПЛИС и микроконтроллеров.

Издание предназначено для инженеров, радио любителей и студентов вузов, желающих позна комиться с новыми классами активных электрон ных компонентов.



На Web-сайте www.dmk.ru вы найдете дополнительный материал – файлы и примеры, расмотренные в книге.

ISBN 5-93700-017-X



www.dmk.ru





